演化计算方法及应用

窦全胜　陈姝颖　著

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书全面概括了用演化方法求解优化问题的一些新方法，重点介绍了进化规划、粒子群优化、微分演化、文化算法和蚁群算法，并阐述了几种新的改进算法，例如，群体启发进化规划方法、模拟退火粒子群优化算法及有分工策略的粒子群优化等，同时就所涉及的算法进行了系统的实验和比较，讨论了不同算法对不同环境的适应能力。

本书可作为从事群体智能、演化计算等领域的研究人员的参考书，对于解决优化问题有一定的参考和应用价值。

图书在版编目（CIP）数据

演化计算方法及应用 / 窦全胜, 陈姝颖 著. —北京：电子工业出版社，2016.1
ISBN 978-7-121-26482-5

Ⅰ. ①演… Ⅱ. ①窦… ②陈… Ⅲ. ①人工神经网络与计算 Ⅳ. ①TP183

中国版本图书馆 CIP 数据核字（2015）第 117530 号

责任编辑：李 敏
特约编辑：刘广钦
印 刷：
装 订：
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编：100036
开 本：720×1000 1/16 印张：11.25 字数：178 千字
版 次：2016 年 1 月第 1 版
印 次：2016 年 1 月第 1 次印刷
定 价：38.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。
服务热线：（010）88258888。

前言

最优化问题是一门既古老又年轻的学科，特别是随着科学技术的发展，许多更加复杂的大规模优化问题不断涌现出来。本书详细阐述了基于演化计算方法的最优化问题求解，主要内容包括：

① 基于遗传算法的优化问题求解；

② 进化规划和群体启发进化规划；

③ 粒子群优化算法和算法改进；

④ 微分演化方法；

⑤ 基于文化算法的约束优化问题求解；

⑥ 蚁群算法；

⑦ 演化算法的应用示例。

本书共 8 章：第 1 章绪论，介绍本书的研究背景，对求解优化问题的数学方法和演化计算方法进行了简要的概括和比较；第 2 章对基于遗传算法的最优化问题求解过程进行了阐述，对以往的一些处理技巧进行了论述；第 3 章对标准进化规划进行了分析，阐述了群体启发进化规划方法，并把它应用于求解高维优化问题；第 4 章介绍了 PSO 算法，并进一步阐述了关于 PSO 方法的两种改进方法；第 5 章介绍了微分演化方法的执行过程；第 6 章概括了用演化计算方法求解约束优化问题的几个策略，着重介绍了几种不可行个体的处理方式，讨论了基于约束与群体的分离策略的文化算法；第 7 章对蚁群算法和蚁群聚类进行介绍；第 8 章列举了几个演化计算方法在实际问题中的应用示例。

本书的工作得到了国家自然基金（61272244、60970088、61175053、61173173）等项目和山东省高校智能信息处理重点实验室（山东工商学院）的资助，在此表示感谢。

此外，还要特别感谢吉林大学的周春光教授，山东工商学院的范辉教授、原达教授、谢青松教授、刘培强教授，感谢他们为本书写作提供的帮助与支持。由于水平有限，书中难免有疏漏和不足之处，敬请读者指正。作者的联系方式为：qsdou@yeah.net。

作者

2015 年 2 月

目录

第 1 章　绪论 1

1.1　最优化问题 2
1.2　求解优化问题的数学方法 4
1.3　求解优化问题的演化计算方法 5

第 2 章　遗传算法 9

2.1　标准遗传算法 10
2.2　编码 12
2.2.1　二进制编码 12
2.2.2　值编码（Value Encoding） 12
2.2.3　互换编码（Permutation Encoding） 13
2.3　遗传算子 14
2.3.1　交叉 14
2.3.2　变异 15
2.3.3　选择 17
2.4　参数控制 18
2.5　模式定理和隐并行性定理 19
2.6　收缩映射原理 21
2.7　小结 24

第 3 章　进化规划 25

3.1　标准进化规划方法 26
3.2　进化策略 28
3.3　概率分析 29
3.4　群体启发进化规划 33
3.4.1　群体启发进化规划算法 33
3.4.2　PHEP 算法验证 35
3.5　用群体启发进化规划求解高维优化问题 40
3.5.1　高维优化 40
3.5.2　实验结果 41
3.6　小结 44

第 4 章　粒子群优化 45

4.1　标准粒子群优化方法 47
4.2　二进制粒子群优化算法 49
4.3　参数设置 56
4.4　粒子轨迹的确定性分析 59
4.5　粒子的分布特征 62
4.6　粒子的聚度 63
4.7　模拟退火粒子群优化方法 66
4.7.1　模拟退火 67
4.7.2　模拟退火粒子群优化 68
4.8　有分工策略的粒子群优化方法 70
4.9　算法测试 73
4.10　动态优化 76
4.10.1　线性模型 76
4.10.2　环形模型 77
4.10.3　随机模型 77
4.10.4　动态优化仿真 78
4.11　小结 83

第 5 章 微分演化 85

5.1 微分演化方法描述 86
5.2 DE 参数的设置 89
5.3 算法仿真 90
5.3.1 低维条件下的仿真结果 90
5.3.2 高维条件下的仿真结果 91
5.4 微分演化粒子群优化 92
5.5 用 DE 确定 PSO 的最佳参数 95
5.6 小结 98

第 6 章 文化算法 99

6.1 约束的处理 101
6.1.1 可行解和不可行解 101
6.1.2 可行个体评价函数 eval_f 的设计 102
6.1.3 不可行个体的处理 103
6.2 文化算法简介 108
6.2.1 文化算法框架 108
6.2.2 信仰空间的约束表达和信仰空间的更新 109
6.2.3 群体空间的演化 113
6.3 算法测试 113
6.4 小结 114

第 7 章 蚁群优化 116

7.1 蚁群优化算法 117
7.2 蚁群聚类 120
7.3 小结 123

第 8 章　应用举例 125

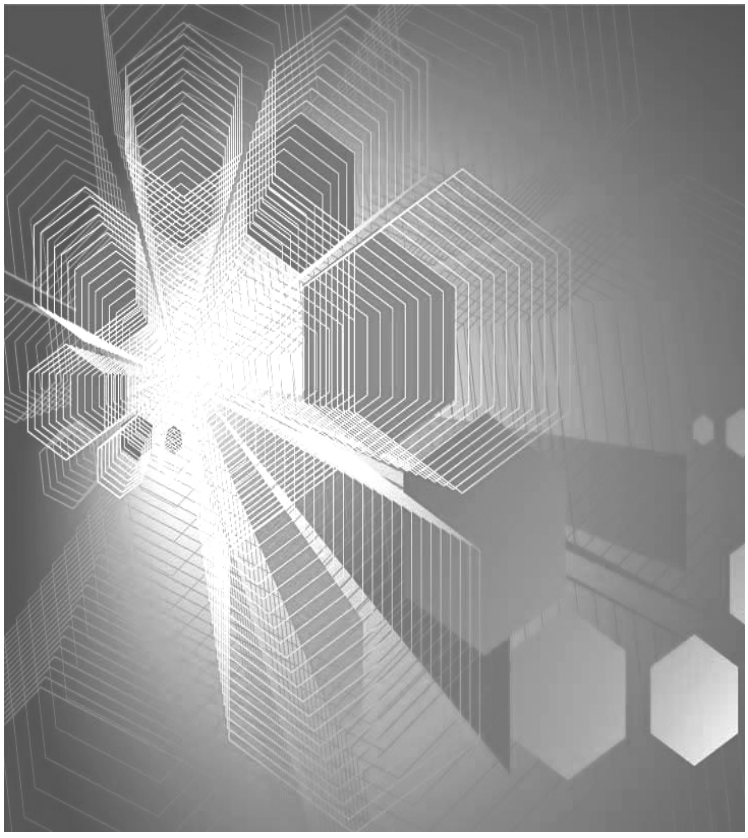
8.1　属性约简 126
8.1.1　信息系统与属性约简 126
8.1.2　常用的属性约简方法 126
8.1.3　基于遗传算法的属性约简 129
8.2　电力负荷关联规则提取 132
8.2.1　问题概述 132
8.2.2　关联规则 133
8.2.3　频繁项集挖掘 136
8.2.4　基于 DPSO 方法负荷规则萃取 138
8.3　神经网络训练 142
8.3.1　神经元模型 143
8.3.2　神经网络 144
8.3.3　神经网络的学习 145
8.3.4　前向神经网络 146
8.4　小结 149

附录 A　无约束优化问题 150

附录 B　约束优化问题 157

参考文献 162

第 1 章　绪论



1.1 最优化问题

最优化是一个重要的数学分支，也是一门应用相当广泛的学科。最优化的目的是对于给出的实际问题，从众多方案中选择出最优方案。例如，工程设计中怎样选择设计参数，使得设计方案既满足设计要求又能降低成本；资源分配中，怎样分配有限的资源，使得分配方案既能满足各方面的基本要求，又能获得好的经济效益。在人类活动的各个领域中，诸如此类，不胜枚举。

应用案例 1：某城市有大型超市 8 个，在“农超对接”对接模式下，8 个超市的某些品种蔬菜全部由该市县级农产品物流中心进行统一配送。其中，县级农产品物流中心位置为 O，8 个超市间距离不同，如表 1-1 所示。配送车辆从 O 出发，选择一个最短的行程对 8 个超市进行农产品配送，最后回到物流中心 O，问如何选择配送路线，使其配送成本最低？

表 1-1　某市 8 个超市间距离　　**单位（km）**

	O	A	B	C	D	E	F	G	H
O	0	25	16	28	19	10	24	20	22
A	25	0	15	9	7	6	23	12	15
B	16	15	0	9.7	4	15	10	13	5.7
C	28	9	9.7	0	12	5.7	6.2	9	12
D	19	7	4	12	0	18	6	16	10
E	10	6	15	5.7	18	0	12	9	15
F	24	23	10	6.2	6	12	0	21	17
G	20	12	13	9	16	9	21	0	12
H	22	15	5.7	12	10	15	17	12	0

应用案例 2：某工厂生产甲、乙、丙、丁共 4 种产品，需要用到 A、B、C 共 3 种原料，每种产品需要使用的各种原料的数量及其可能获得的利润如表 1-2 所示。另外，A、B 两种原料供应量有限，单位生产周期内只能提供一定的数量，而 C 种原料一经开包使用就必须用足一定量后才可停止使用，且不能单独使用。现有关数据均见表 1-2，问应如何安排生产，才能使该厂所获利润达到最大值?

表 1-2　加工产品所需原料及可能获得的利润

原料	加工每件产品所需原料				单位周期内原料的供应量或必须使用量
	甲	乙	丙	丁	
A	1.0	1.2	1.4	1.5	≤2100
B	0.5	0.6	0.6	0.8	≤1000
C	0.7	0.7	0.8	0.8	≤1300
每件利润	12	15	8	10	

类似这样的优化问题在生产生活中大量存在，其研究历史可以追溯到十分古老的极值问题。早在 17 世纪，英国伟大科学家牛顿发明微积分的时代，已经提出极值问题，后来又出现 Lagrangian 乘数法。1847 年，法国数学家 Cauchy 研究了函数沿什么方向下降最快的问题，提出了最速下降法。1939 年，前苏联数学家 Л.В.Канторович 提出了解决下料问题和运输问题这两种线性规划问题的求解方法。人们关于最优化问题的研究工作，随着历史的发展不断深入。但是，任何学科的进步，都受到历史条件的限制，直至 20 世纪 30 年代，最优化这个古老的课题还未形成独立的学科。在最近的二三十年中，伴随着计算机技术的高速发展和优化计算方法的进步，各种优化问题的理论研究发展迅速，新方法不断出现，实际应用日益广泛。在计算机的推动下，一些超大规模的优化问题得以实现，最优化理论与方法在经济规划、工程设计、生产管理、交通运输等方面得到了广泛应用，成为一门十分活跃的学科。

最优化问题的一般形式为[1]

$$\begin{aligned} &\text{Min } f(x) \\ &\text{s.t. } x \in X \end{aligned} \tag{1-1}$$

其中，$x \in R^n$ 是决策变量，$f(x)$ 为目标函数，$X \subset R^n$ 为约束集或可行域。特别地，如果约束集 $X = R^n$，则最优化问题（1-1）称为无约束最优化问题。在本书中，若 X 为 R^n 中的超立方体，也近似地把问题（1-1）看成无约束最优化问题。

优化问题通常带有一些约束条件，称为约束最优化问题，表述为

$$\begin{aligned} &\text{Min } f(x) \\ &\text{s.t. } c_i(x) = 0,\ i \in E \\ &\quad\ c_j(x) \geqslant 0,\ j \in I \end{aligned} \tag{1-2}$$

这里，E 和 I 分别是等式约束指标集和不等式约束指标集，$c_i(x)$ 是约束函数。当目标函数和约束函数均为线性函数时，问题称为线性规划；当目标函数和约束函数中至少有一个是变量 x 的非线性函数时，问题称为非线性规划。此外，根据决策变量、目标函数和要求的不同，最优化还可以分为整数规划、动态规划、网络规划、非光滑规划、随机规划、几何规划、多目标规划等若干分支。

1.2 求解优化问题的数学方法

求解最优化问题通常采用迭代方法，其基本思想如下：给定一个初值点 $x_0 \in R^n$，按照某一迭代规则产生一个序列 $\{x_k\}$，使得当 $\{x_k\}$ 为有穷点列时，其最后一个点是最优化问题的最优解。当 $\{x_k\}$ 为无穷列时，它有极限点，且其极限点是最优化问题的最优解。一个好的算法应该具备如下特征：迭代点能够稳定地接近局部极小点 x^* 的邻域，然后迅速收敛于 x^*。迭代方法的基本结构如下[1]。

步骤 1 给定初始点 x_0，确定搜索方向 d_k。

步骤 2 确定步长因子 a_k。

步骤 3 令 $x_{k+1} = x_k + a_k d_k$。 （1-3）

步骤 4 若 x_{k+1} 满足某种终止条件，则停止迭代，得到近似最优解。否则，重复上述步骤。

从式（1-3）中可以看出，迭代算法的核心如下：构造适应各种问题的步长因子 a_k 和搜索方向 d_k，不同的 a_k 和 d_k 的构造方式，形成了不同的优化算法。

最速下降法是求解无约束优化问题最经典的方法，又称梯度下降法，该方法以函数在 x_k 处的负梯度方向作为算法的下降方向，最速下降法在最优化中具有重要的理论意义。但是，最速下降方向仅反映了被优化函数的局部性质，对于许多问题，最速下降法并非“最速下降”，而是下降非常缓慢；利用函数的二阶导数信息，即在椭球范数 $\|\cdot\|_{G_k}$ 下，取负梯度方向，最速下降法就变成了所谓的牛顿法，牛顿法利用了函数二阶导数信息，为克服牛顿法中 Hesse 矩阵未必正定的难题，Goldfeld 等人提出了修正牛顿法。类似的

方法还有有限差分牛顿法、负曲率方向法、不精确牛顿法等。牛顿法的基本思想是在迭代点 x_k 附近用二次函数逼近 $f(x)$，但这种方法只能保证算法的局部收敛性，信赖域方法是一种能够保证算法总体收敛的方法，并且能够解决 Hesse 矩阵不正定和 x_k 为鞍点等困难，这类方法包括 Levenberg-Marquardt 方法、双折线步法（The Double Dogleg Step Method）等。另一种介于最速下降法和牛顿法之间的方法是共轭方向法，这种方法只需要一阶导数信息，既克服了最速下降法收敛慢的缺点，又避免了存储和计算牛顿法所需的二阶导数信息。共轭方向法是从研究二次函数的极小化问题中产生的，但是可以推广到处理非二次函数极小化问题，典型的共轭方向法包括共轭梯度法和拟牛顿法等。

求解约束优化问题的典型方法有罚函数法、可行方向法、逐步二次规划法等。罚函数法的主要思想是：构造具有“惩罚”性质的函数，对不可行点进行惩罚，借助罚函数把约束问题转化成无约束问题。有许多罚函数被广泛使用，如内点罚函数、乘子罚函数、光滑精确罚函数和非光滑精确罚函数等。典型的可行方向法包括可行点法、广义消去法、广义概约梯度法、投影梯度法等。逐步二次规划法包括 Lagrange-Newton 法、Wilson-Han-Powell 法等。

以上方法的详细描述可以在文献[1]～[3]中找到，用数学方法求解优化问题的历史相对悠久，当前仍然在不断的发展过程中，这些传统方法大多是针对某些特定问题，并且对搜索空间要求相对严格，有些方法还要使用被优化函数的各阶导数信息。但是，随着科学和技术的发展，优化问题也变得异常复杂，有的问题甚至无法用函数关系来表达，对于这类问题，采用上述方法，不可能得到圆满的结果。因此，需要进一步研究和探索新的优化思想和优化方法。

1.3　求解优化问题的演化计算方法

传统演化计算（Evolutionary Computation）是模拟“物竞天泽”与“自然遗传”的生物进化过程所产生的随机化计算模型[4]。它的起源可以追溯到 20 世纪 50 年代，其中有影响的工作是 Bremermann[5]、Friedberg[6][7]、Box[8]等。但是，几乎在以后的 30 年时

间里，这一领域的工作对于广大科学工作者来说还是相当陌生，造成这种状况的主要原因是：当时缺少强大的计算机硬件平台和早期演化计算本身的方法缺陷。一直到 20 世纪 70 年代，Holland[9]、Rechenberg[10]、Schwefel[11]、Fogel[12]等人的奠基性工作才慢慢地改变了这种状况。特别是近十年，以演化计算为主题的学术活动逐年增加，如今面向应用的演化计算研究几乎渗透到各行各业[13]~[15]。演化计算技术在各方面得以广泛应用的原因如下：首先是得益于计算机性能的极大提高；其次演化计算具有自组织性、自学习性、自优化等智能特征，同时又具有内在并行性、原理的简单性、优良的全局性、应用的广泛性等特点。

传统演化计算是由 3 个相互联系、但事实上又彼此独立发展起来的 3 个分支组成，它们是遗传算法（Genetic Algorithms，GAs）、进化规划（Evolutionary Programming，EP）、进化策略（Evolutionary Strategies，ES）。遗传算法最初是由 Holland[9][16][17]等作为研究自适应过程的一般模型提出的，后来经 De Jong[18]~[21]、Goldberg[22]~[26]等人的进一步扩展和完善，现在主要应用于优化领域。进化规划是由 Fogel L.J.[12][27]作为产生人工智能的一种尝试而提出的，Burgin[28][29]、Atmar[30]等人在这一领域作了深入的工作，其方法是演化一个有限状态机（Finite State Machines，FSM），使之具有最佳的预测能力。后来，Fogel D.B[31]~[33]等借助于进化策略方法对进化规划进行了发展，并应用于数值优化等问题中。Rechenberg[34][35]和 Schwefel[36][37]为求解多参数优化问题提出了进化策略（Evolutionary Strategies，ES）。后来，其他学者在这个领域继续研究，从原始的 $(1+1)-\mathrm{ES}$ 进化策略，发展到 $(m+1)-\mathrm{ES}$ 进化策略，进而发展到 $(u+l)-\mathrm{ES}$ 进化策略。从 20 世纪 80 年代开始，由于计算机硬件性能的提高，使得演化计算可应用于求解高度复杂的现实优化问题。

传统的演化计算包括遗传算法、进化策略和进化规划。20 世纪 90 年代以来，遗传程序设计（Genetic Progromming，GP）作为遗传算法的一个分支独立发展起来，并在自动程序设计方面显示了初步潜力，因此，有些学者把演化计算扩展为包括遗传程序设计在内的 4 个分支。一些研究[38]把演化计算进一步扩充为“仿生”类演化计算和“拟物”类演化计算，其中“仿生”类演化计算包括上述 4 个分支，“拟物”类演化计算包括模

拟物理系统演化和社会系统演化等所有算法模型，如 Boltzmann 演化策略、文化算法（Culture Algorithms，CAs）、粒子群优化等（Particle Swarm Optimization，PSO）等。有学者将演化计算称为进化计算，本书用演化计算来概括所有用计算机系统模拟大自然演化过程来求解现实问题的一切算法，既包括所有“仿生”类方法，又包括“拟物”类算法。

随着各个学科研究的不断深入，出现了越来越多的难以用传统方法解决的实际问题，人们开始尝试使用演化计算方法。与传统数学方法相比，演化计算方法有如下特点。

（1）演化计算的处理对象可以是参数本身，也可以是经过某种映射形成的特定编码，编码形式可以是矩阵、树、图、集合、串、序列、链和表等。这个特点使演化计算有广泛的应用领域。

（2）演化计算采用群体搜索策略，而传统方法多采用单点搜索策略，这种特点使演化计算具有极好的全局性，减少陷入局部最优的风险；同时，也使演化计算本身易于大规模并行实现，可充分发挥高性能计算机系统的作用。

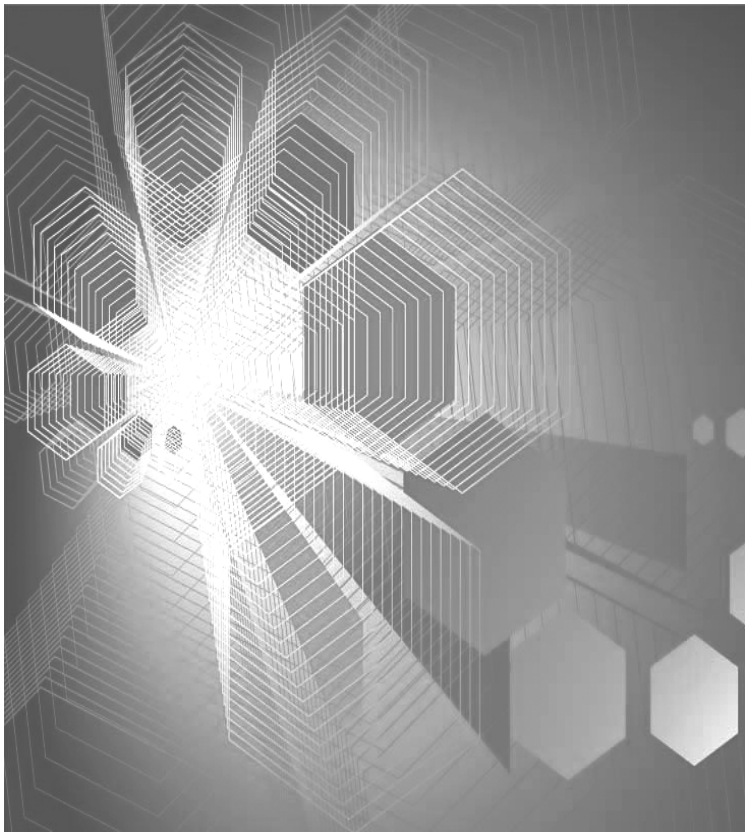
（3）演化计算基本不依赖搜索空间的知识及其他辅助信息，它采用适应度函数来评价个体，并在此基础上驱动演化过程，而对适应度函数本身无特别严格要求。传统方法则要求函数有诸如连续、可微或空间凸性等条件。这使演化计算有更广泛的应用范围；演化计算用概率的变迁规则来控制搜索的方向，从表面上看好像是在盲目搜索，实际上它遵守某种随机概率，在概率意义上朝最优解方向靠近，因此，它不同于通常采用确定性规则的传统方法，需要构造合适的下降方向。

与求解优化问题的数学方法相比，演化计算方法的优势如表 1-3 所示。通过表 1-3 的比较可以看出，在解决某些复杂优化问题上，演化计算方法有着得天独厚的优势，本书将对演化计算中的几个典型算法进行阐述、比较和分析。

表 1-3　用数学方法和演化方法求解优化问题比较

方　法	数学方法	演化计算方法
应用范围	往往是面向问题的	独立于问题
对初值的依赖性	对初值有很强的依赖性，即初值敏感性	基于群体操作，基本不依赖初值的选取
对搜索空间和优化函数的要求	一般要求优化函数连续、各阶导数等信息，对搜索空间也有严格要求	不用搜索空间等辅助信息
计算复杂性	计算复杂	计算简单

第 2 章　遗传算法



20 世纪 60 年代，Rechenberg I[34]第一次在算法中引入了“进化”的思想，这一思想逐渐被其他一些研究者发展。1975 年，Holland[16]首次提出遗传算法（Genetic Algorithms，GA），之后 Holland 和他的学生及同事又对这一算法进行了丰富和补充。遗传算法是对生物遗传学和自然选择机理的仿真，并通过人工方式构造的一类搜索算法，从某种程度上说，遗传算法是对生物进化过程的数学描述。

生物种群的生存过程普遍遵循达尔文进化准则，群体中个体根据对环境的适应能力而被自然界所选择或淘汰。进化过程的结果反映在个体的结构上，体现了个体外部特性与内部机理间的逻辑关系。通过个体之间的交叉、变异来适应大自然环境。生物染色体用数学方式或计算机方式来体现，就形成一串数码，这里仍称之为染色体，染色体的适应度值是对应于每个染色体的一个实数，这个值决定了染色体最终将被选择，还是被淘汰。

遗传算法自提出以来，在国际上已经形成了一个比较活跃的研究领域。目前，遗传算法已广泛应用于函数优化、排序问题、人工神经网络训练、分类系统、计算机图像处理和机器人运动规划等领域。作为演化计算技术的一部分，遗传算法依然在不断地向前发展。本章将对遗传算法及其理论进行概括，详细阐述遗传算法求解最优化问题的过程和最新进展。

2.1 标准遗传算法

遗传算法类似于自然进化，通过作用于染色体上的基因，寻找好的染色体来求解问题。与自然界相似，遗传算法对求解问题本身一无所知，它所需要的仅是对算法所产生的每个染色体进行评价，并基于适应值来选择染色体，使适应性好的染色体有更多的繁殖机会。在遗传算法中，通过随机方式产生初始群体；通过适应度函数给每个个体一个数值评价，淘汰低适应度的个体，选择高适应度的个体参加遗传操作，经过遗传操作后的个体集合形成下一代新种群。遗传算法借用了许多生物遗传学中的术语，其中，“个体（Individual）”表示问题的一个潜在解；“种群（Population）”表示一组个体的集合。

在标准遗传算法中，个体用一定长度的二进制编码来表示，每个个体的性能用“适应度函数”来度量。遗传算法中的“遗传操作（Genetic Operator）”主要有“变异”、“交叉”和“选择”。下面给出标准遗传算法的形式化定义。

定义个体 $p \in B^l$，其中，$B^l = \{0,1\}^l$ 代表所有长度为 l 的二进制编码组成的集合。种群 $P=\{p_1, p_2, \cdots, p_n\}$ 为 n 个个体的集合，$P \in B^{l,n}$。定义适应度函数 $f: B^l \to R$，$f(p)$ 代表个体 p 的优劣程度。变异操作定义为算子 $m: B^l \to B^l$；交叉操作定义为算子 $c: B^l \times B^l \to B^l \times B^l$；选择操作定义为算子 $s: B^{l,n} \to B^{l,n}$。定义 $P_m \in [0,1]$ 为变异概率，表示个体发生变异的概率；定义 $P_c \in [0,1]$ 为交叉概率，表示交叉算子作用于个体的概率。那么，遗传算法就可以由七元组（P，f，m，c，s，P_m，P_c）来表示。算法 2-1 所示是遗传算法的基本执行过程。

算法 2-1　标准遗传算法	
0	Algorithm: Genetic Algorithms，GA
1	初始化群体；
2	计算群体上每个个体的适应度值；
3	按由个体适应度值所决定的某个规则选择将进入下一代的个体;
4	按概率 P_c 进行交叉操作；
5	按概率 P_m 进行突变操作；
6	若满足某种停止条件，则执行 Step7，否则执行 Step2；
7	输出种群中适应度值最优的染色体作为问题的满意解或最优解。

一般情况下，算法的终止条件包括：

① 完成了预先给定的进化代数；

② 种群中的最优个体在连续若干代没有改进或平均适应度在连续若干代基本没有改进；

③ 所求问题最优解的达到了某一精度。

2.2 编码

许多实际问题的潜在解可以化为简单的位串形式，将被求解问题的潜在解变换为位串或其他形式表示的过程称为编码；反之，将位串或其他形式的编码变换为原问题解的过程称为译码。

2.2.1 二进制编码

二进制编码是最常见的一种编码方法，遗传算法最早采用的就是二进制编码，在使用二进制编码时，每一个染色体就是一个由 0 或者 1 组成的字符串。例如：

染色体 A：

101100101100101011100101

染色体 B：

111111100000110000011111

采用二进制编码，染色体的长度依赖于问题所要求的精度。其缺点如下：对于很多问题的表达都很不自然，并且用于求解多维、高精度问题时，会遇到一些障碍。假设问题的维度为 100，搜索区域为[-500,500]，要求精度为 10^{-6}，对于这一问题，用二进制表达，染色体的长度是 3000，同时会产生 101000 的搜索空间，搜索效果不理想。因此，在解决一些实际问题时，其他形式的编码也被广泛应用。

2.2.2 值编码（Value Encoding）

在很多问题中，二进制编码并不能确切地表达问题的潜在解，因此，可以采用所谓的值编码。在值编码中，每个染色体就是一串取值。这些取值可以是与问题有关的任何值：整数、实数、字符或者其他一些更复杂的符号。例如：

染色体 A：

1.2324-5.3243-0.4556-2.3293-2.4545

染色体 B：

ABDJEIFJDHDIERJFDLDF

染色体 C：

Back-Back-Right-Forward-Left

值编码对于一些特殊的问题是很有效的。但是，对于这种编码方法，它的遗传算子就要按照问题和编码方法具体去设计。

2.2.3　互换编码（Permutation Encoding）

互换编码可以用来解决排序问题，如旅行商问题（Travelling Salesman Problem，TSP）和任务排序问题（Task Ordering Problems，TOP）。采用互换型编码的染色体如下所示。

染色体 A：

1-5-3-2-6-4-7-9-8

染色体 B：

8-5-6-7-2-3-1-4-9

互换编码对排序问题非常适用，在旅行商问题中，每个城市可以用一个数字表示，搜索空间是 n 个城市的排列集合，采用互换编码能够最贴切地表述上述问题。

在求解最优化问题上，二进制编码和值编码中的实数编码是最常用的两种编码形式，并且在实际应用中，值编码的使用呈上升趋势。这种趋势使二进制编码遗传算法的理论研究者感到惊讶[23][16]，因为一些分析分析表明：使用低基数性的字母表进行编码，可以增强模式的处理，而来自实际运算的结果似乎直接反驳了这种观点。实数编码在实际运算中能够得到更好的结果。

2.3 遗传算子

遗传操作主要有 3 种：选择（Selection）、交叉（Crossover）、变异（Mutation）。改进的遗传算法大量扩充了遗传操作，这里不一一说明。在解决优化问题上，二进制编码和实数编码是最常用的两种编码方式，本节只讨论基于这两种编码的各种遗传操作。一般情况下，二进制编码的遗传算子，经过适当改进，都可应用于其他编码形式的染色体上。

2.3.1 交叉

交叉操作是指将在群体中按某种方式选择出的两个个体 P1 和 P2 作为父个体，并把这两个个体的部分码值进行交换的过程。以下 1～4 和 5～7 分别是基于二进制编码和实数编码的交叉操作。

1．**单点交叉**（Single Point Crossover）

在两个父个体上选择一个交叉点，子代在交叉点前面的基因从一个父个体那里得到，后面的部分从另外一个父个体那里得到，如下所示。

父代	子代
1100｜1011	11011011
+	
1101｜1111	11001111

2．**双点交叉**（Two Point Crossover）

选择两个交叉点，子代基因在两个交叉点间部分来自一个父个体，其余部分来自另外一个父个体，如下所示。

父代	子代
10｜0010｜10	11001011
+	
11｜0111｜11	10011110

3．**均匀交叉**（Uniform Crossover）

子代基因的每一个位随机地来自两个父代基因中的一个，这种交叉算子产生的新染色体是无法确定的。

4．**算术交叉**（Arithmetic Crossover）

对父个体做某种代数运算，从而产生一个新的个体。下面就是使用“和”运算（AND）产生新染色体的例子。

父代　　子代

11001011

+（AND）　　10001010

10011110

5．**实数算术交叉**[39][40]（Arithmetical Crossover）

设 X_1 和 X_2 为两个父染色体，实数算术交叉产生的两个子染色体为 $X_1^{'}=aX_1+(1-a)X_2$ 和 $X_2^{'}=\alpha X_2+(1-\alpha)X_1$。若两个父染色体 $X_1=(x_1, \cdots, x_q)$，$X_2=(y_1, \cdots, y_q)$，交叉后产生的两个子个体分别是 $Y_1^{'}=(\alpha x_1+(1-\alpha)y_1, \cdots, \alpha x_q+(1-\alpha)y_q)$ 和 $Y_2^{'}=(\alpha y_1+(1-\alpha)x_1, \cdots, \alpha y_q+(1-\alpha)x_q)$。当 $a=0.5$ 时，实数算术交叉又被称为保证平均交叉[40]（Guaranteed Average Crossover）。

6．**简单交叉**（Simple Crossover）

设 X_1 和 X_2 为两个父染色体，且 $X_1=(x_1, \cdots, x_q)$，$X_2=(y_1, \cdots, y_q)$。设简单交叉操作发生在两个染色体的第 k 位上，则产生的后代为 $X_1^{'}=(x_1, \cdots, x_k, y_{k+1}, \cdots, y_q)$ 和 $Y_1^{'}=(y_1, \cdots, y_k, x_{k+1}, \cdots, x_q)$。

2.3.2　变异

在二进制编码中，变异算子的主要作用是弥补进化过程中群体多样性的损失，通过变异算法能够开辟新的搜索领域；在实数编码时，变异的作用更加重要，下面介绍几个实数编码中常见的变异算子。

1．均匀变异

设染色体 $X=(x_1,\mathbf{L},x_k,\mathbf{L},x_q)$，$k\in\{1,\mathbf{L},q\}$，且分量 x_k 定义区间是 $[a_k,b_k]$，变异算子作用于该染色体，并产生新的染色体 $X'=\{x_1,\mathbf{L},x_k',\mathbf{L},x_q\}$，其中，$x_k'$ 是 $[a_k,b_k]$ 里满足正态分布的随机数。

2．非均匀变异

在传统的遗传算法中，算子的作用和迭代次数是没有直接关系的。从而，当算法演化到一定代以后，由于缺乏局部搜索，传统的遗传算子将很难获得收益。基于以上原因，Z. Michalewicz 首先将变异算子的结果与演化代数联系起来，使得在演化初期，变异算子范围相对较大，随着演化的推进，变异的范围越来越小，起着一种对演化系统的微调作用。其具体描述如下。

设 $X=(x_1,\mathbf{L},x_q)$是一个父染色体，分量 x_k 被选为演化个体进行变异，其定义区间是 $[a_k,b_k]$，则变异后的解为 $X'=(x_1,\mathbf{L},x_{k-1},x_k',\mathbf{L},x_q)$，其中

$$x_k'=\begin{cases}x_k+\Delta(t,b_k-x_k) & ,\ \mathrm{rnd}(1)=0\\ x_k-\Delta(t,x_k-a_k) & ,\ \mathrm{rnd}(1)=1\end{cases}$$

上式中的 rnd(1) 表示在集合{0,1}产生均匀分布随机数的函数。

函数 $\Delta(t,y)$ 的具体表达式为

$$\Delta(t,y)=y\left(1-r^{\left(1-\frac{t}{T}\right)}\right)$$

其中，r 是[0, 1]上的一个随机数，T 为最大代数，λ 是决定非一致性程度的一个参数，它起着调整局部搜索区域的作用，其取值一般为 2～5。

3．边界变异

边界变异本质上是均匀变异的一个变种，在均匀变异中，x_k' 是区间 $[a_k,b_k]$ 上满足正态分布的随机数，而在边界变异中 x_k' 以相同的概率要么取 a_k，要么取 b_k。当最优解位于或靠近可行搜索空间的边界时，边界变异非常适用。

2.3.3　选择

选择操作是根据个体的适应度决定它在下一代中被淘汰还是被保留的过程。一般，根据达尔文的进化论，选择将使适应度较大的（优良）个体有较大的生存机会，而适应度较小（低劣）的个体继续生存的机会也较小。选择染色体以适应环境的方法有很多，常用的选择算子有赌盘选择[41]（Roulette）、联赛选择[42]（Tournament），此外还有几何分布选择[41]（Geometic Distribution）、排序选择[43][44]（Rank-based Model）、最优保存策略[45]（Elitist Model）、确定式采样选择[46]（Deterministic Sampling）、无回放随机选择[46]（Remainder Stochastic Sampling with Replacement）、期待值选择[45]（Expected Value Model）、随机联赛选择[42]（Stochastic Tournament Model）等。下面对遗传算法常用的几种选择方法进行介绍。

1．**赌盘选择**（Roulette Wheel Selection）

令 f_i 表示种群中第 i 个染色体的适应度值，$\sum f_i$ 表示群体的适应度值之总和，由此，染色体 i 存活到下一代的概率为 $P_i = f_i / \sum f_i$ 。染色体适应度越大，它被选择到的机会就越大。更加直观地，可以把这种选择机制想象成在一个轮盘赌的机器上放置了种群中所有的染色体。每一个染色体所占的面积大小和它的适应度成正比，如图 2-1 所示。

显然，在赌盘选择中，适应度越大的染色体被选到的机会就越大。

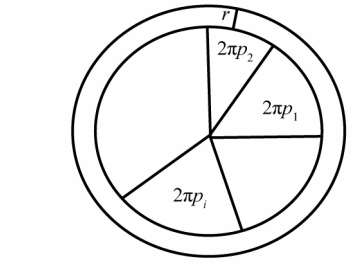


图 2-1　轮盘赌选择

2．**排序选择**（Rank Selection）

赌盘选择方法很简单，但是当适应度变化较大时就会有问题发生。例如，当种群中最好的染色体的适应度在“赌盘”上占很大比例时，其他染色体被选择到的机会就会很

小。分级选择把种群分几成个等级，这样，所有染色体都有机会被选择到。这种选择方式是基于这样一个认识：算法过早收敛的主要原因是在群体中有“超级”个体存在，这些超级个体的适应度要比群体平均适应度大很多，以至于它们有大量的后代，由于群体规模是固定的，它们会阻止其他个体的子代在下一代中出现，在迭代过程中，“超级”个体可能会排除其他有希望的染色体，并造成算法快速收敛到局部最优点。排序选择使最好的个体与其他个体的被选中的概率缩小了，缓解了群体的选择压力，但这种策略会导致算法收敛速度受到一定影响。

3．**期待值选择**（Expected Value Model）

这种选择方法为每个个体 X_i 引入一个数 μ，初始时 $\mu = f(X_i)/\bar{f}$，其中，$\bar{f} = \left[\sum_{i=1}^{q} f(X_i)\right]/q$，$q$ 是群体规模。该染色体被选中时，μ 就相应地减去一个数，当 $\mu<0$ 时，这个染色体将被淘汰。

4．**精英模型**（Elitist Model）

精英模型是最早出现的选择策略，它强制保存最好的染色体。这种选择方式可以使算法较快地收敛，但是，强制保存最好的染色体往往会造成群体的多样性遭到破坏，导致“早熟”现象的发生。

2.4 参数控制

遗传算法在执行过程中，有如下几个重要的参数需要考虑。

1．**群体规模**

种群规模一般要根据实际问题而定，如果种群规模太小，种群中就没有充分的多样性，不利于适应度高的染色体的进化，致使算法得不到满意的解。可以认为大的种群规模有利于种群进化和健壮发展。但如果个体过多，每进行一轮进化花费的机器时间就多，致使算法的效率低。根据实际问题，群体规模一般取 20～100 的一个数。

2．**杂交概率（P_c）**

当杂交概率（P_c）较大时，优秀个体出现的概率大，收敛速度较快，但是频繁的新旧替换容易破坏群体中的优良模式，对进化运算产生不利影响。当 P_c 取值比较小时，产生新的个体速度较慢，群体的收敛速度会受到影响。一般情况下 P_c 的取值为 0.4～0.99。

3．**变异概率（P_m）**

当变异概率 P_m 较大时，会造成优良个体的破坏，使得遗传算法近似于随机搜索。当变异概率较小时，算法产生新个体和抑制早熟的能力就较差。一般情况下 P_m 的取值为 0.0001～0.1。

遗传算法参数的设置目前尚无规律可循，主要依靠经验和试凑的方法得到。在实际运用中，采用综合优选参数和变参数法可以获得较为满意的运行参数。

2.5　模式定理和隐并行性定理

一些研究者对遗传算法的运行机理进行了研究[47][48]，Radolph 证明了一般遗传算法不一定收敛，只有每代保存了最优个体时，算法才会收敛到全局最优[48]。这一结论的证明是通过对遗传算法构造马尔可夫（Markov）链完成的。当遗传算法收敛时，求得的解通常只是所要解决问题的一个近似解或满意解。从数学分析的角度来看，收敛过程是一个无限逼近过程，而计算过程是一个有限自动机，因此，通过遗传算法程序求得的解总是一个近似解。近似解与问题真正的最优解的差是一个统计意义下的量，也就是说每次程序运行得到解的质量可能有较大的差别。遗传算法是一个以适应度为依据，通过对种群中个体进行遗传操作来实现种群内个体结构重组的迭代处理过程。它具有传统优化算法（如梯度算法等）所不具备的稳健性、自适应性和全局优化性等特点。下面的模式定理和算法的隐并行性论述进一步揭示了算法的内在本质。

定义 2-1　模式：基于三值字符集{0,1,*}所产生的能描述具有某些结构相似性的 0、1 字符串集的字符串被称为模式。

以长度为 5 的串为例，模式*101*描述了在位置 2、3、4 具有形式“101”的所有字符串。通配符“*”可以取 0 或 1，它是一个描述符，仅仅为描述上的方便而引入的符号，并非是遗传操作中实际的运算符号。

通过模式概念的引入，可以看到，一个串实际上隐含着多个模式，而一个模式又可以隐含在多个子串中。因此，通过分析模式在遗传操作下的变化，可以了解什么性质被延续，什么性质被抛弃，从而把握遗传算法的实质。为进一步区分不同模式的特性，以下引入模式阶与模式长度的概念。

定义 2-2　模式阶：模式 H 中的确定位置的个数称为该模式的阶，记为 $O(H)$。

定义 2-3　模式长度：模式 H 中的第一个确定位置和最后一个确定位置之间的距离称为该模式的定义长度，记为 $d(H)$。

定理 2-1　模式定理[49]**：**在遗传算子选择、交叉和变异的作用下，具有低阶、短定义长度并且平均适应度高于群体平均适应度的模式在子代中将得以指数级增长。

模式定理给出了经过遗传算子作用后，某一模式的样本在下一代中得以生存的数目的下界。同时，模式定理也指出，由于交叉算子的作用，并非所有的模式都能被高效地处理，那些具有较长定义长度的模式将遭到破坏。Holland 和 Goldberg 指出了遗传算法能够有效处理的模式个数的下界。

定理 2-2　隐并行性定理[49]**：**遗传算法有效处理的模式个数为 $O(n^3)$个。即尽管遗传算法每一代仅对 n 个个体进行运算，但却隐含地处理了 $O(n^3)$个模式。

对于遗传算法是否能最终获得全局最优解，即算法是否收敛，Holland 提出了一种积木块（Building Block）假说，即低阶、短定义长度、高平均适应度的模式（积木块），在遗传算子作用下相互结合，能生成高阶、长定义长度、高平均适应度的模式，并可能最终生成全局最优解。令人遗憾的是，此结论并没有得到证明。

2.6　收缩映射原理

遗传算法收敛性研究在演化计算领域中是一个很重要且极具挑战性的理论问题，本节通过 Bannach 不动点理论[50]来刻画遗传算法的收敛性。

定义 2-4　度量空间：用 R 表示实数集合，S 为一非空集合，$d: S\times S\to R$ 为一映射，对 $\forall x, y\in S$，满足如下条件：

①$d(x,y)\geqslant 0$，若 $d(x,y)=0$，当且仅当 $x=y$；

②$d(x,y)=d(y,x)$；

③$d(x,y)+d(y,z)\geqslant d(x,z)$。

则集合 S 和映射 d 构成了度量空间，记作 $\langle S,d\rangle$，d 被称为距离。

设 $\langle S,d\rangle$ 为一度量空间，$f: S\to S$ 为一映射，称映射 f 是收缩的，当且仅当 $\exists e\in[0,1)$，对所有 $x, y\in S$，$d(f(x), f(y))\leqslant d(x,y)$ 都成立。

定义 2-5　柯西序列：称序列 $\{p_n\}$ 为度量空间 $\langle S,d\rangle$ 上的一个柯西序列，若对 $\forall e>0$，都存在一个整数 k，当 $m,n>k$ 时，$d(p_m, p_n)<e$。

若对度量空间 $\langle S,d\rangle$ 上的所有柯西序列 $\{p_n\}$ 都有 $p=\lim\limits_{n\to\infty} p_n$，且 $p\in S$，则称量度空间 $\langle S,d\rangle$ 是完备的。

定理 2-3　收缩映射原理：设 $\langle S,d\rangle$ 是一完备的度量空间，$f: S\to S$ 为一收缩映射，则 f 有一个唯一的不动点 $x\in S$，使得对任意 $x_0\in S$ 都有

$$x=\lim_{i\to\infty} f^i(x_0)$$

其中，$f^0(x_0)=x_0$，$f^{i+1}(x_0)=f\left[f^i(x_0)\right]$。

Banach 不动点理论处理了距离空间上的收缩映射问题，它证明了这样的映射 f 有唯一的不动点，即仅有一个 x 使 $f(x)=x$。不动点理论被作为定义计算语义学的强大工具而被广泛接受。经过分析，会发现度量空间的概念提供了一个非常简单而且自然的表达遗传算法语义学的方法，遗传算法能被定义成群体间形状的变化，收缩映射原理也就

能够很自然地应用于遗传算法，即如果构造度量空间 S，使其成员由群体构成，那么，根据 Banach 定理，任何收缩映射 f 都有唯一的不动点，该不动点是通过 f 应用于一个任选的初始群体 $P(0)$迭代而获得的。这样，如果发现一个合适的度量空间，并使此空间中的群体操作是收缩的，那么，算法的收敛性自然得到了保证，而这种收敛性与初始群体的选择是无关的。

假定，群体规模 pop_size=n 是固定的，即群体 $P=\{X_1,\mathbf{L},X_n\}$，定义整个群体的评价函数为

$$\mathrm{Eval}(P)=\frac{1}{n}\sum_{X_i\in P}\mathrm{Eval}(X_i)$$

其中，$\mathrm{Eval}(X_i)$ 是群体 P 中第 i 个染色体的适应度。集合 S 由所有可能的群体 P 组成。定义映射 $d:S\times S\to R$ 满足

$$d(P_1,P_2)=\begin{cases}0 & P_1=P_2\\ \left|1+M-\mathrm{Eval}(P_1)\right|+\left|1+M-\mathrm{Eval}(P_2)\right| & P_1\neq P_2\end{cases}$$

其中，M 为域中所有个体评价值的上限，即对所有个体 X，$\sum_{X_i\in P}\mathrm{Eval}(X)\leqslant M$。故对所有可能的群体 P，Eval(P)$\leqslant nM$。容易验证如下结论。

① 对所有的群体 P_1 和 P_2，$\delta(P_1,P_2)\geqslant 0$；当且仅当 $P_1=P_2$ 时，$\delta(P_1,P_2)=0$。

② $d(P_1,P_2)=d(P_2,P_1)$。

③ $d(P_1,P_2)+d(P_2,P_3)\geqslant d(P_1,P_3)$。

因此，$\langle S,d\rangle$ 为一度量空间。值得注意的是，对于遗传算法而言，主要处理的是有限度量空间，这样度量空间 $\langle S,d\rangle$ 可以看成完备的。用 f 来表示遗传算法中交叉、变异和选择等所有算子的复合，从而，第 t+1 代群体可以表示为 $P(t+1)=f(P(t))$，显然 f 每作用于群体一次，会产生一个新的群体。对 f 加以如下限制：若产生的新群体适应度比原群体的适应度有所改进，则把新群体作为下一代群体保留下来；否则，把原群体作为下一代保留下来。这样 f 就可以看成一个收缩映射。在这些条件限制下的遗传算法能满

足收缩映射原理的要求。很明显，如果迭代 $f:P(t)\to P(t+1)$ 改变了群体适应度，即如果

$$\mathrm{Eval}\left(P_1(t)\right)<\mathrm{Eval}\left(f\left(P_1(t)\right)\right)=\mathrm{Eval}\left(P_1(t+1)\right)$$

并且

$$\mathrm{Eval}\left(P_2(t)\right)<\mathrm{Eval}\left(f\left(P_2(t)\right)\right)=\mathrm{Eval}\left(P_2(t+1)\right)$$

那么

$$\begin{aligned}d\left(f\left(P_1(t)\right),f\left(P_2(t)\right)\right)&=\left|1+M-\mathrm{Eval}\left(f\left(P_1(t)\right)\right)\right|+\left|1+M-\mathrm{Eval}\left(f\left(P_2(t)\right)\right)\right|\\&<\left|1+M-\mathrm{Eval}\left(P_1(t+1)\right)\right|+\left|1+M-\mathrm{Eval}\left(P_2(t+1)\right)\right|\\&=d\left(P_1(t+1),P_2(t+1)\right)\end{aligned}$$

由此，根据收缩映射原理，有 $P^*=\lim_{i\to\infty}f^i\left(P(0)\right)$。即修正后的遗传算法收敛到群体 P^*，它是群体空间中唯一的不动点。

显然， P^* 表示的是一个包含全局最优的群体，注意到

$$\mathrm{Eval}(P)=\frac{1}{n}\sum_{X_i\in P}\mathrm{Eval}(X_i)$$

这就意味着，当该群体中所有个体都有相同的值（全局最优）时，即得到不动点 P^*，并且 P^* 与初始群体无关。

这种按上述原理修正后的遗传算法，称为收缩映射遗传算法。具体过程如算法 2-2 所示。

算法 2-2　收缩映射遗传算法（CM-GA）

0	Algorithm: Compression mapping Genetic Algorithms，CM-GA
1	初始化群体和参数，令 t=0；
2	计算群体 $P(t)$中每个个体的适应度值，从而计算出群体的适应度值 Eval($P(t)$)；
3	令 $P(\mathrm{temp})=f\left(P(t)\right)$；
4	若 $\mathrm{Eval}\left(P(\mathrm{temp})\right)\geqslant\mathrm{Eval}\left(P(t)\right)$，则 $P(t+1)=P(\mathrm{temp})$，否则 $P(t+1)=P(t)$；
5	若满足某种停止条件，则执行步骤 6，否则，执行 $t=t+1$，执行步骤 2；
6	输出种群中适应度值最优的染色体作为问题的满意解或最优解。

以上算法满足压缩映射原理的假设，因此，能够收敛到群体空间中唯一的不动点 P^*。但存在这样一个问题，就是没有考虑所求问题存在多个全局最优的情况，在这种情况下，适应度最大值不止一个，对于最优群体 P_1、P_2，算法中的群体作用函数 f 不是真正意义上的收缩映射。事实上，算法将要收敛到其中的一个最优群体上[50]。

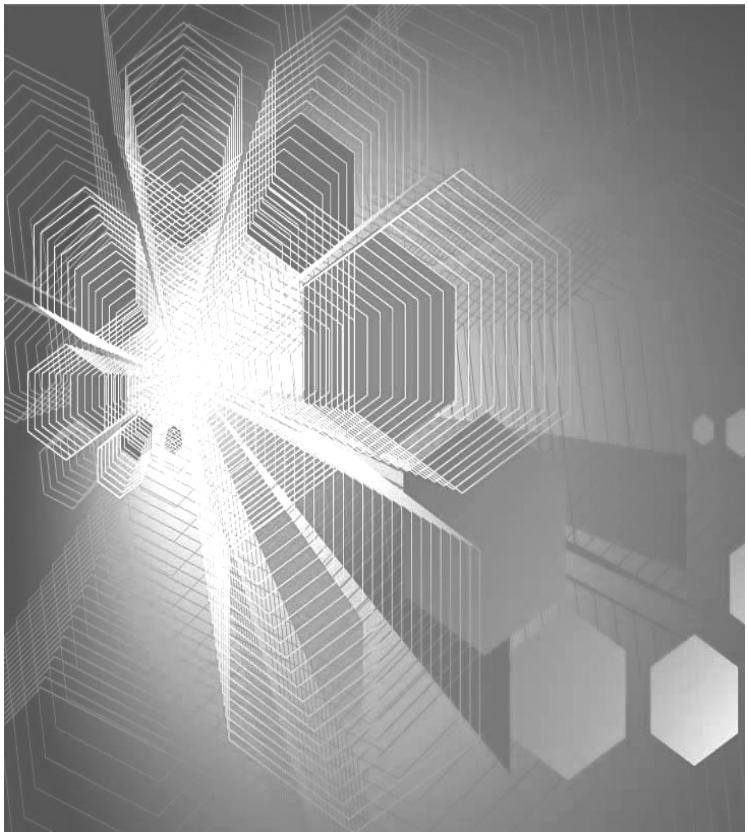
对于收缩映射遗传算法，初始群体的选择只是影响收敛的速度。这种算法的结果也是在无限时间域里收敛到全局最优，在算法的执行过程中，会存在这样的阶段：算法长时间找不到可以接受的群体，这种情况的发生与求解问题的难度及算法本身参数的设置都有很大关系，这里不再讨论。

2.7 小结

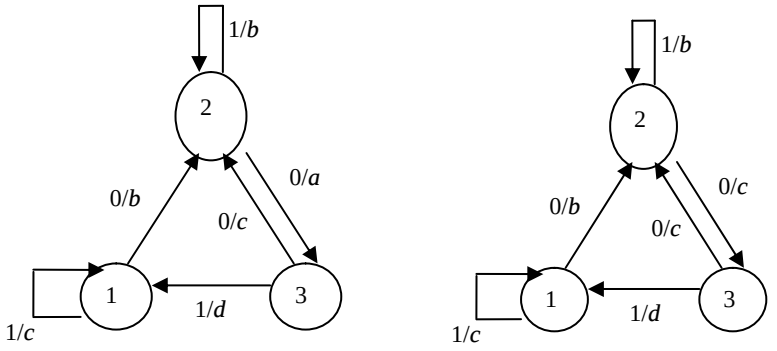
遗传算法作为一种对生物进化现象的仿真程序，取得了人工遗传的模拟效果，在遗传算法中，遗传操作和编码机制是两个重要的因素，本章着重介绍了遗传算法的编码机制、各种遗传算子和参数控制，同时，对遗传算法的一些理论加以阐述。在实际应用中，如果一个问题不能求得目标函数的全局最优值，而只能或只希望求一定意义下的“满意解”，这时，可供选择的方法之一自然是遗传算法，因为遗传算法比其他算法有更多的优势。可喜的是，近年来遗传算法在商业应用方面取得了一系列重要成果。这是推动遗传算法不断发展的另一个原因。

遗传算法具有隐并行性，它可容易地改造成并行、分布式算法，用来解决那些比较复杂的问题。遗传算法的理论和机理仍不是很清楚，这可能与生命科学的研究一样，将是一个永恒的研究课题，目前，已有很多学者对遗传算法理论作了一些深入的研究，近十几年来，基于遗传算法的研究和应用已相当多，本章仅加以概括和整理，并未作深入的探讨。

第 3 章　进化规划



进化规划（Evolutionary Programming，EP）技术是由 Lawrence Fogel[51]最早提出的，起初的目的是针对人工智能的演化过程，原始 EP 方法中的个体采用有限状态机（Finite State Machines，FSM）来表达，每个有限状态机表示该问题的一个潜在解，即表示的是一个特殊的行为。进化规划首先产生一个有限状态机的群体，每个亲体通过变异产生单个后代，如图 3-1 所示，有 5 种经常使用的变异算子：输出符号的改变、状态转换、状态加入、状态删除和初始状态改变。这些变异以某种概率被挑选，从变异后的群体中选择最好的个体保留到下一代群体中。随着研究的不断深入，进化规划技术逐渐推广到数值优化领域[51]~[55]，并且与进化策略（Evolutionary Strategy，ES）、遗传算法等技术融合交叉。本章对求解优化问题的进化规划进行了分析，阐述基于群体启发的进化规划方法，并把它应用于求解高维优化问题。



（a）变异前　　（b）变异后

图 3-1　有限状态机

3.1　标准进化规划方法

近几年，进化规划术的研究有了很大的进展，很多优秀的方法成功地应用于各类优化问题。在应用于优化问题上，进化规划与二进制编码遗传算法相比较主要有以下两点不同。

1．个体的表达方式不同

经典的遗传算法通常采用二进制向量来表示个体，而进化规划则用实值向量来表示个体。

2．演化算子不同

在进化的过程中，进化规划主要采用的是变异操作，而在遗传算法中变异操作往往被忽略。

相比之下，进化规划与实数编码的遗传算法有很多相似之处，两种方法的改进版本在技术细节上有许多交叉和相融之处。在求解优化问题的进化规划方法中 CEP（Classical Evolutionary Programming）、FEP（Fast Evolutionary Programming）、IFEP（Improved Fast Evolutionary Programming）、REP（Robust Evolutionary Programming）等方法先后在文献[56]～[61]中被提出和应用。算法 3-1 所示是 CEP 的执行过程。

算法 3-1　经典进化规划算法

0 Algorithm：Classical Evolutionary Programming

1 在解域 $S \subseteq R^n$ 内随机的产生由 μ 个个体组成的群体，每个个体由一对实值向量构成：(x_i, h_i)，$\forall i \in (1, \mathbf{L}, m)$，其中，$x_i$ 是目标变量，h_i 是变异向量；

2 根据评价函数 $f(x)$计算每个个体的评价函数值；

3 对群体中的每一个个体(x_i, h_i)进行变异，通过如下变异产生一个后代（$x_i^{'}, h_i^{'}$），$i = 1, \mathbf{L}, m$，有

$$\begin{aligned} x_i^{'}(j) &= x_i(j) + h_i(j) N_j(0,1) \\ h_i^{'}(j) &= h_i(j) \exp(t^{'} N(0,1) + t N_j(0,1)) \end{aligned} \qquad (3\text{-}1)$$

其中，$x_i(j)$、$x_i^{'}(j)$、$h_i(j)$、$h_i^{'}(j)$ 分别表示向量 x_i、$x_i^{'}$、h_i、$h_i^{'}$ 的第 j 个分量，$N(0,1)$为满足正态分布的随机数，τ 和 $\tau^{'}$ 通常被设置成$(\sqrt{2\sqrt{n}})^{-1}$和$(\sqrt{2n})^{-1}$；

4 计算每一个后代（$x_i^{'}, h_i^{'}$）（i=1, $\mathbf{L}$, μ）的评价函数值；

5 所有父个体和子个体组成一个由 2μ 个个体组成的群体。对于该群体中的每一个个体，从群体中随机等概率地选取出 q 个个体和它进行比较，在每次比较中，如果该个体的评价值不大于与之比较的个体的评价值（对于所研究的最小化问题），则称该个体获得一次胜利，其中 q 为事先指定的正整数；

6 从 2μ 个个体中选择获胜次数最多的 μ 个个体作为下一代的父个体；

7 若满足终止条件，则算法停止，否则，转到步骤 3。

FEP 执行步骤与 CEP 相同，只是用 Cauchy 分布函数代替 CEP 中的 Gaussan 分布函数[56]，以增大跳出局部最优的概率。IFEP 同时使用了 Cauchy 分布函数和 Gaussan 分布函数进行变异操作，并选出最好的个体。进化规划技术在发展过程中，不断地与其他演化类方法融合，其中一个重要的方法就是进化策略（Evolutionary Strategies，ES），下面对进化策略进行阐述。

3.2 进化策略

进化策略是一类仿效自然界进化规律，解决参数优化问题的方法[4][39]。1963 年，柏林大学的 Ingo Rechenberg 和 Hans Paul Schwefel 为解决风洞试验中的参数优化问题提出了该方法。进化策略方法中的染色体采用浮点数表达，进化过程中只采用变异算子。在群体中每个个体被表达成一对浮点向量，如 $\boldsymbol{v}=(\boldsymbol{x},s)$，其中，向量 $\boldsymbol{x}$ 是搜索空间中的一个点，s 是标准偏差向量，个体的变异是通过公式 $\boldsymbol{x}(t+1)=\boldsymbol{x}(t)+N(0,s)$ 实现，其中 $N(0,s)$ 为一个标准偏差为 s 的随机 Gauss 向量，这种变异与自然界的观察是一致的：一般情况下，自然界中的物种，较小的变异比较大的变异更经常发生，后代当且仅当较好地适应环境，并且满足所有约束时，才被接受为群体中的一个新成员。早期的进化策略只有一个成员，它通过变异产生一个后代，后代和父代竞争，差的个体被淘汰。这种单成员进化策略被精练的表述为(1+1)-ES。

对进化策略收敛率的优化，Rechenberg 提出了“1/5 成功法则[4]”：对于所有变异，变异成功的概率应保持在 1/5，若变异成功概率大于 1/5，则增加变异算子的方差；反之，缩小其方差。即

$$h(t+1)=\begin{cases} c_d h(t) & ,\ P(k)<1/5 \\ c_i h(t) & ,\ P(k)>1/5 \\ h(t) & ,\ P(k)=1/5 \end{cases} \qquad (3\text{-}2)$$

其中，$h(t)$ 为群体第 t 代变异方差向量，$P(k)$ 为群体最近 k 代变异成功的概率。$c_d<1$ 和 $c_i>1$ 调整变异方差的增加和减少。在实际计算中，Schwefel 建议使用 $c_i=1.22$、$c_d=0.82$，

但这种经验并不绝对，这些参数的设置是与实际问题相关的，对于不同的问题要进行不同的修正。

为改善进化策略的执行性能，很自然会想到适当地增大群体规模，由此多成员进化策略[39]被提出，并表示为 $(m+1)-\mathrm{ES}$，其中，m 是群体规模。

多成员进化策略进一步发展成熟，即变成 $(m+l)-\mathrm{ES}$ 和 $(m,l)-\mathrm{ES}$，这些策略的主要思想是，允许变异方差的自适应调整，而不是用一些确定性的算法来改变它们的值。$(m+l)-\mathrm{ES}$ 是对多成员进化策略 $(m+1)-\mathrm{ES}$ 的一个自然扩展，其中，m 个个体生成 l 个后代。新的 $(m+l)$ 个个体通过再次选择，重新生成由 m 个个体构成的群体。在 $(m,l)-\mathrm{ES}$ 中，m 个个体生成 l 个后代，并且 $l>m$，从 l 个后代中选择出 m 个个体构成新一代群体。这样，每一个个体的生命被限制在一代里。

关于实数编码遗传算法，进化策略和进化规划的比较是十分细腻的，它们在提出的初期，有着不同的背景，但随着各种方法的不断发展，它们之间不断地相互借鉴、相互融合、相互交叉。本书认为，这 3 种方法本质上应该归于一类。因此，在细节的表述上只是顺应于某种习惯，并未作严格的区分。

3.3　概率分析

设 $f:S\to R$，$S\subset R^n$，则用演化方法求解最优化问题（1-1），应满足以下条件：

① 搜索空间 S 为 Lebesgue 可测，f 在 S 上有界；

② f 在搜索空间 S 上存在全局最优解；

③ 用 $m(S)$ 表示 S 的 Lebesgue 测度，对于 $\forall x>0$，$m(\{x\mid f(x)\leqslant \min\limits_{y\in S} f(y)+x\})>0$。

从标准进化规划（Classical Evolutionary Programming，CEP）中可知，个体按式（3-1）进行更新，其中，$N(0,1)$是满足正态分布的随机数。把个体在每一代的修正量表示为 $D=hN(0,1)$，h 为变异向量，从而可以近似地认为 $D=N(0,h)$，其密度函数可以表示为

$$f_{G(0,h^2)}(x)=\frac{1}{h\sqrt{2\pi}}\exp\left(-\frac{x^2}{2h^2}\right),-\infty<x<+\infty \qquad (3\text{-}3)$$

假设 x^* 为所求问题的一个全局最优点，初始点位置为 0（可以通过适当的平移）。由此，个体按式（3-1）变异后，进入 x^* 的某一邻域 (x^*-e, x^*+e) 的概率为

$$P_{G(0,h^2)}(|x-x^*|\leqslant e)=\int_{x^*-e}^{x^*+e} f_{G(0,h^2)}(x)\mathrm{d}x \tag{3-4}$$

其中，e 为一个正数，反映了领域的尺寸，由积分中值定理，存在一个正数 d，且 $0<d<2e$，如图 3-2 所示，使得

$$\int_{x^*-e}^{x^*+e} f_{G(0,h^2)}(x)\mathrm{d}x=2e\, f_{G(0,h^2)}(x^*-e+d) \tag{3-5}$$

从而有

$$\frac{\partial}{\partial h}P_{G(0,h^2)}\left(\left|x-x^*\right|\leqslant e\right)=\frac{\partial}{\partial h}\int_{x^*-e}^{x^*+e} f_{G(0,h^2)}(x)\mathrm{d}x=\frac{\partial}{\partial h}2e\, f_{G(0,h^2)}(x^*-e+d)$$

$$=2e\left(\frac{(x^*-e+d)^2}{h^4\sqrt{2\pi}}\exp\left(-\frac{(x^*-e+d)^2}{2h^2}\right)-\frac{1}{h^2\sqrt{2\pi}}\exp\left(-\frac{(x^*-e+d)^2}{2h^2}\right)\right)$$

$$=\frac{2e}{h^2\sqrt{2\pi}}\exp\left(-\frac{(x^*-e+d)^2}{2h^2}\right)\left(\frac{(x^*-e+d)^2}{h^2}-1\right)$$

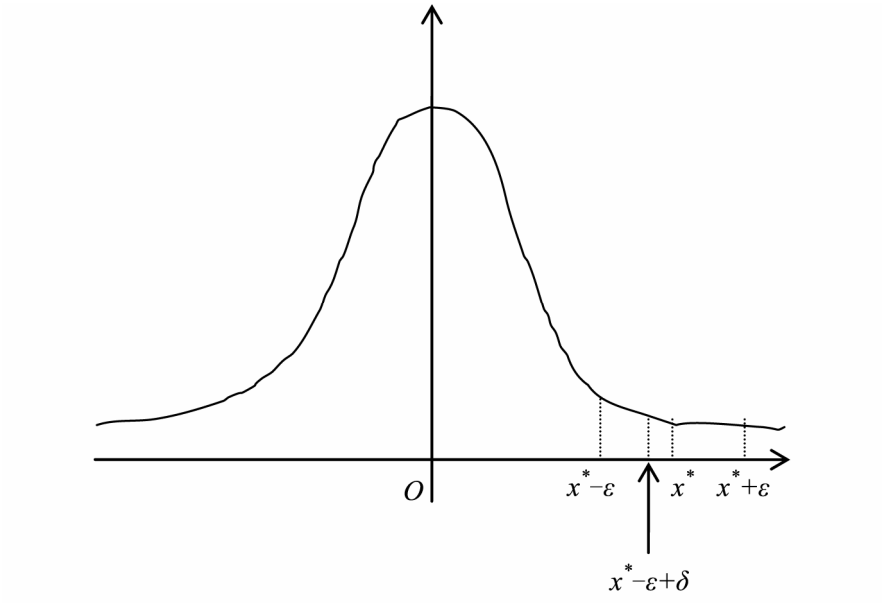


图 3-2　变异成功概率

显然，当$\frac{\left(x^*-e+d\right)^2}{h^2}-1>0$时，即$h<\left|x^*-e+d\right|$时，有

$$\frac{\partial}{\partial h}P_{G\left(0,h^2\right)}(|x-x^*|\leqslant e)>0 \tag{3-6}$$

个体变异后，进入x^*的某一邻域(x^*-e,x^*+e)的概率$P(|x-x^*|\leqslant e)$单调递增；反之，当$h>|x^*-e+d|$时，概率$P(|x-x^*|\leqslant e)$单调递减。

通过上述推导可以得出：若变异向量满足正态分布，当$h<|x^*-e+d|$时，变异步长h越大，个体进入邻域(x^*-e,x^*+e)的概率就越大；而当$h>|x^*-e+d|$时，变异步长h越大，个体进入邻域(x^*-e,x^*+e)的概率就越小。

在进化规划中，变异算子另一个常用的分布函数是 Cauchy 分布，其密度函数为

$$f_{\text{Cauchy}}(x)=\frac{t^2}{\pi\left(t^2+x^2\right)} \tag{3-7}$$

这里，参数t与正态分布中的方差h的含义相似，在算法中起着变异步长的作用。同样采用上述分析方法，有

$$\frac{\partial}{\partial t}P_{\text{Cauchy}}\left(\left|x-x^*\right|\leqslant e\right)=\frac{\partial}{\partial t}\int_{x^*-e}^{x^*+e}f_{\text{Cauchy}}(x)\mathrm{d}x$$

由中值定理，得

$$\begin{aligned}\frac{\partial}{\partial t}P_{\text{Cauchy}}\left(\left|x-x^*\right|\leqslant e\right)&=\frac{\partial}{\partial t}\left(2e\,f_{\text{Cauchy}}\left(x^*-e+d\right)\right)\\&=\frac{2e}{\pi}\frac{\partial}{\partial t}\left(\frac{t}{t^2+\left(x^*-e+d\right)^2}\right)\\&=\frac{2e}{\pi}\frac{\partial}{\partial t}\left(\frac{1}{t^2+\left(x^*-e+d\right)^2}-\frac{2t^2}{\left(t^2+\left(x^*-e+d\right)^2\right)^2}\right)\\&=\frac{2e}{\pi}\frac{\left(x^*-e+d\right)^2-t^2}{\left(t^2+\left(x^*-e+d\right)^2\right)^2}\end{aligned}$$

上式所得结果与正态分布相似，当$t<|x^*-e+d|$时，变异步长h越大，个体进入邻域(x^*-e,x^*+e)的概率就越大；而当$t>|x^*-e+d|$时，变异步长h越大，个体进入邻域(x^*-e,x^*+e)的概率就越小。

假设x^*是求解问题的全局最优点，即对$\forall x\in S$，$f(x^*)\leqslant f(x)$，且x^*不是孤立点（假设 3 保证了x^*不是孤立点）。否则，无论方差如何选择，从S中的任何一点出发，经过变异撞到x^*的概率均为 0。

进一步分析从S中的任何一点x_0出发，经过变异撞进x^*的邻域$W=(x^*-e,x^*+e)$的概率$P\{x_0\to x\,|\,x\in W\}$。为简化问题，不妨设$x_0=0$，则

$$P\{x_0\to x\,|\,x\in W\}=\int_a^b p(t)\,\mathrm{d}t \tag{3-8}$$

其中，$p(t)$是概率密度函数，$a=x^*-e$，$b=x^*+e$。

若变异采用正态分布，则式（3-8）可写成

$$\begin{aligned}
&P\{x_0\to x\,|\,x\in W\}=\int_a^b f_{G(0,h^2)}(t)\,\mathrm{d}t\\
&=\frac{1}{\sqrt{2\pi}h}\int_a^b\exp\left(-\frac{t^2}{2h^2}\right)\mathrm{d}t\\
&=\frac{1}{\sqrt{2\pi}h}\int_a^b\left(1+\left(-\frac{t^2}{2h^2}\right)+\frac{1}{2!}\left(-\frac{t^2}{2h^2}\right)^2+\mathbf{L}+\frac{1}{n!}\left(-\frac{t^2}{2h^2}\right)^n+\mathbf{L}\right)\mathrm{d}t\\
&=\frac{2d}{\sqrt{2\pi}h}+\frac{1}{\sqrt{2\pi}h}\sum_{n=1}^{+\infty}\left[\frac{1}{(2n+1)n!}\left(\frac{-1}{2h^2}\right)^n\left(b^{2n+1}-a^{2n+1}\right)\right]
\end{aligned}$$

因此，$P\{x_0\to x\,|\,x\in W\}$可以由下式给出估计值：

$$P\{x_0\to x\,|\,x\in W\}=\frac{2d}{\sqrt{2\pi}h}+\frac{1}{\sqrt{2\pi}h}\sum_{n=1}^{N}\left[\frac{1}{(2n+1)n!}\left(\frac{-1}{2h^2}\right)^n\left(b^{2n+1}-a^{2n+1}\right)\right]$$

，式中N的取值决定了估计值的精度。

3.4　群体启发进化规划

算法早熟是进化规划遇到的最大问题，算法早熟的原因很多，求解问题的复杂性和欺骗性是造成算法早熟的一个原因，从问题入手，对问题进行改造和变换是一种解决方法，但这需要对问题有深入的理解，而且有些实际问题复杂的根本无法进行处理，这方面本书不作更多的讨论。我们仅从进化规划算法本身出发，通过改进算法的选择机制和变异机制，保持群体多样性，克服早熟。

3.4.1　群体启发进化规划算法

在个体变异过程中，新个体由式（3-1）产生，式中的变异向量h_i起着“步长”的作用，当$|h|$的值较大时，个体的搜索范围较大，有很强的“开拓”能力，相反，当$|h|$值较小时，个体的搜索范围变小，局部“探索”能力得到增强。通常情况下，h有几种自适应方式，有的与迭代的次数n有关，或者与变异成功的概率有关。一些方法采用了 ES 方法的 1/5 成功法则，取得了较好的效果。但是，在算法执行的后期，变异成功的难度不断增大，变异成功的概率不断减小。因此，算法在后来的执行过程中很难跳出局部最优的吸引，造成“早熟”；另一方面，进化规划算法多是按某一概率进行选择个体来组成下一代群体，这样使评价值较高的个体及其后代存活下来的概率远大于评价函数较低的个体，从而使群体的多样性在一定程度上遭到破坏，加快了“早熟”的发生。

群体启发进化规划方法（Population Heuristic Evolutionary Programming，PHEP）通过改进进化规划的选择策略和变异策略，在试图保证算法“局部搜索”能力的同时保持群体的多样性，PHEP 方法的主要思想是：设x_i（i=1, $\mathbf{L}$,μ）是第t代群体 POP(t)中的元素，我们按如下方式选择其中元素构成第$t+1$代群体 POP(t+1)，计算 $f_{\text{avg}}=\sum f(x_i)/m$，把 POP($t$)中所有满足 $f(x_i)<f_{\text{avg}}$（$i=1,\mathbf{L},m$）的x_i构成的集合记为P_1，P_1中元素的个数记为$\text{Count}(P_1)$，然后对所有$y_j\in \text{POP}(t)-P_1$（$j=1,\mathbf{L},m-\text{Count}(P_1)$）进行变异，设$\hat{y}_j$是$y_j$变异后产生的新个体，把所有满足 $f(\hat{y}_j)\leqslant f(y_j)$ 的$\hat{y}_j$构成的集合记为 P_2，若$\text{Count}(P_1\mathbf{U}P_2)<m$，则从集合$P_1\mathbf{U}P_2$中随机选择$m-\text{Count}(P_1\mathbf{U}P_2)$个个体进行变异，设$\hat{z}_k$是集合$P_1\mathbf{U}P_2$中个体$z_k$变异后产生的新个体，把所有满足 $f(\hat{z}_k)\leqslant f(z_k)$ 的$\hat{z}_k$构成的集

合记为 P_3，若 $\mathrm{Count}(P_1 \mathbf{U} P_2 \mathbf{U} P_3) < m$，在 $P_1 \mathbf{U} P_2 \mathbf{U} P_3$ 中随机选择 $m - \mathrm{Count}(P_1 \mathbf{U} P_2 \mathbf{U} P_3)$ 个个体直接构成 P_4。PHEP 通过 4 个集合元素的个数占个体总数的比例来掌握群体中个体的分布情况，进而调整变异向量的值。下一代群体 $\mathrm{POP}(t+1)$ 等于上述 4 个集合的并集或者保持不变。群体启发进化规划的详细执行过程如算法 3-2 所述。由算法步骤 4 可知，当 $O_{\mathrm{direct\_sel}} > l$，且变异步长较小时，说明群体的变异成功率很低，且群体多样性很差，算法有早熟的趋势。此时，如果一味地缩小变异向量，会造成群体的过早收敛，因此，算法增大变向量，以跳出局部最优的吸引，并把群体直接保留到下一代中；由步骤 5，当 $O_{\mathrm{normal}} + O_{\mathrm{mut}} > g$ 时，表明群体中只有少数较差的个体，且变异成功率较高，这时增大变异向量，以提高算法效率，当 $n < O_{\mathrm{normal}} + O_{\mathrm{mut}} < a$ 时，说明算法变异成功率降低，应该适当缩小变异向量，提高算法变异成功率；由步骤 6，当 $O_{\mathrm{normal}} + O_{\mathrm{mut}} < n$ 时，说明群体中存在少数个体，它们的评价函数值远小于其他个体（对于最小优化问题），此时，当集合 P_2 变异成功率较高时，增大变异向量，反之，变异向量不变。

算法 3-2　群体启发进化规划

0 Algorithm：Population Heuristic Evolutionary Programming, PHEP

1 令 $k=1$，在解域 $S \subseteq R^n$ 内随机的产生由 m 个个体组成的群体 $P(k)$，每个个体由一对实值向量构成：$(x_i, h_i)\ \forall i \in (1, \mathbf{L}, m)$，其中，$x_i$ 是目标变量，h_i 是变异向量。其中，$P(t)$ 表示第 t 代群体。

2 利用指定的评价函数 f 计算：$f_{\mathrm{avg}} = \left(\sum_{i=1}^{m} f(x_i) \right) / u$。

3 用 $\hat{P}_{\mathrm{avg}}(k)$ 表示 $P(k)$ 中所有满足如下条件：$f(x_i) < f_{\mathrm{avg}}$，$\forall i \in (1, \mathbf{L}, m)$ 的个体的集合。用如下方法构造集合 $P_{\mathrm{pre\_sel}}$：(a)令 $P_{\mathrm{pre\_sel}}$ 为空，若 $x_j \in \hat{P}_{\mathrm{avg}}(k)$，$j = 1, \mathbf{L}, m$，其中 $m < m$，则 x_j 加入到 $P_{\mathrm{pre\_sel}}$，并记 $O_{\mathrm{normal}} = m / m$；(b)规定变异次数，对所有 $x_l \in P(k) - \hat{P}_{\mathrm{avg}}(k)$，$l = 1, \mathbf{L}, n-m$ 按如下方法进行变异：$x_l' = x_l + h_l N(0,1)$，得到 x_l'，若 $f(x_l') < f(x_l)$，则加 x_l' 到 $P_{\mathrm{pre\_sel}}$，设满足这一条件的 x_l' 有 n 个，记 $O_{\mathrm{mut}} = n / m$；(c)若 $m + n < \mu$，则在 $P_{\mathrm{pre\_sel}}$ 中随机地选择 $\mu - m - n$ 个个体 x_p，$p = 1, \mathbf{L}, m-m-n$，规定变异次数，变异得到 x_p'，若 $f(x_p') < f_{\mathrm{avg}}$，则保留 x_p' 到 $P_{\mathrm{pre\_sel}}$，设满足这一条件的 x_p' 有 q 个，记 $O_{\mathrm{sel\_mut}} = q / m$；(d)若 $m + n + q < \mu$，则在 $P_{\mathrm{pre\_sel}}$ 中随机地选择 $m - m - n - q$ 个个体补充到 $P_{\mathrm{pre\_sel}}$ 中，记 $O_{\mathrm{direct\_sel}} = (m - m - n - q) / m$。

4 如果 $O_{\mathrm{direct\_sel}} > l$，并且变异向量 h_i 小于某规定值，则作如下调整：$h_i = h_i t_l$，其中 l、t_l 为大于 1 的正常数。$P(k+1) = P(k)$，转到 Step7。

续表

5 若 $O_{\text{normal}}+O_{\text{mut}}>g$ ，则调整变异向量 $h_i=h_i t_g$，若 $n<O_{\text{normal}}+O_{\text{mut}}<a$ ，则调整变异向量 $h_i=h_i t_a$，其中 g 、a 为小于 1 的正数，$0<a<g$ ，$t_g>1$ ，$0<t_a<1$。$P(k+1)=P_{\text{pre\_sel}}$，转到 Step7。

6 在 $O_{\text{normal}}+O_{\text{mut}}<n$ 的条件下，若 $O_{\text{sel\_mut}}>b$，并且变异向量 h_i 小于某规定值，则作如下调整：$h_i=h_i t_b$，其中 $t_b>1$，否则保持原变异向量不变。

7 满足终止条件，过程结束。否则，k=k+1，转到 Step2。

3.4.2　PHEP 算法验证

从附录中选择 f_1～ f_{13}，搜索空间维度如表 3-1 所示。

表 3-1　测试问题及维度

单峰函数（Unimodal Function）		多峰函数（Multimodal Function）	
函数	维数（n）	函数	维数（n）
f_1	30	f_8	30
f_2	30	f_9	30
f_3	30	f_{10}	30
f_4	30	f_{11}	30
f_5	30	f_{12}	30
f_6	30	f_{13}	30
f_7	30		

其中，f_1～f_7 为单峰函数，f_8～f_{13} 为多峰函数。多峰函数有许多局部最优点，因此，搜索其全局最优非常困难。如图 3-3 所示为二维空间下，函数 f_8（广义 Schwefel 问题）的图像。当维数增加时，最优点的个数呈指数增加。

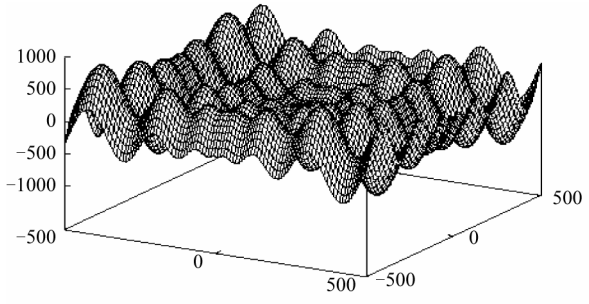
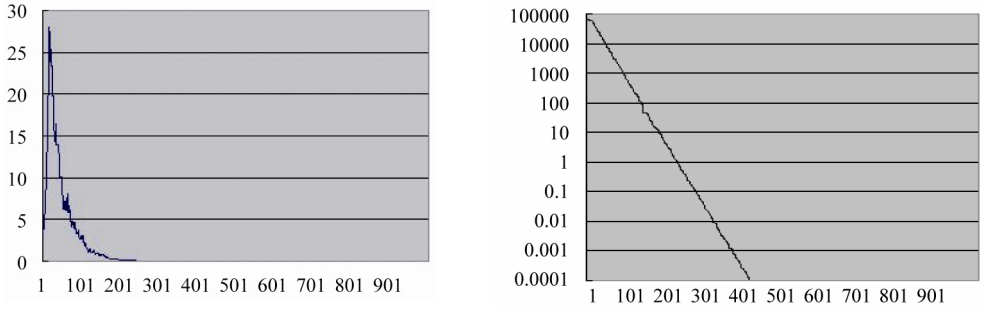


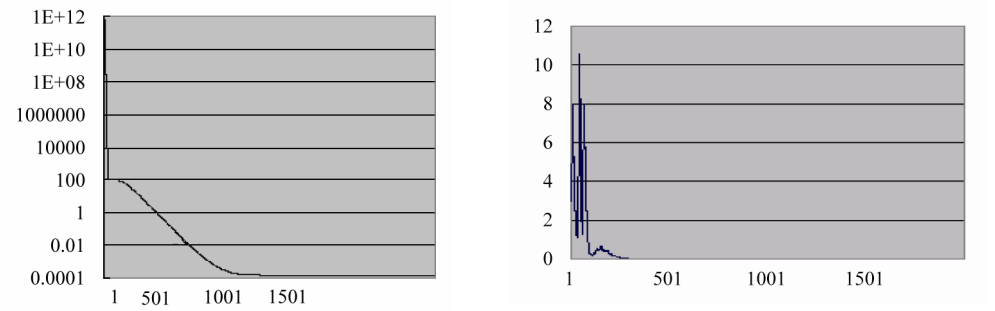
图 3-3　广义Schwefel问题

用 f_1～f_7 来考查算法收敛的速率，取 $m=100$、$l=0.3$、$t_l=1.05$、$g=0.8$、$t_g=1.15$、$a=0.65$、$t_a=0.85$、$n=0.3$、$b=0.45$、$t_b=1.05$，PHEP 运行 50 次，所得最优解平均值曲线和变异向量变化曲线如图 3-4～图 3-10 所示。



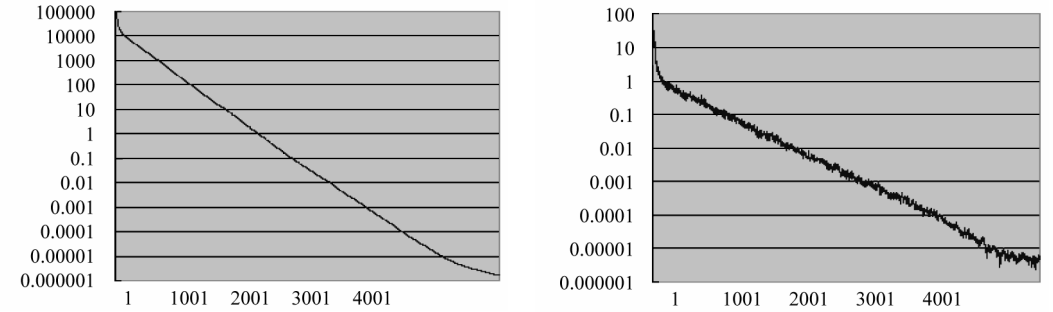
（a）　　（b）

图 3-4　f_1：球体模型



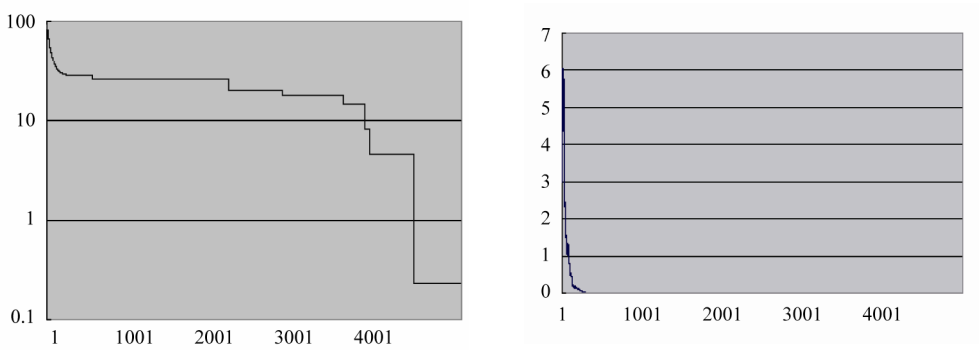
（a）　　（b）

图 3-5　f_2：Schwefel问题 2.22



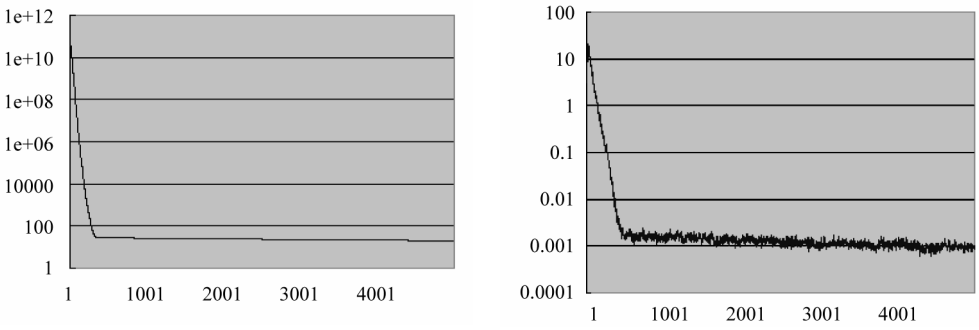
（a）　　（b）

图 3-6　f_3：Schwefel问题 1.2



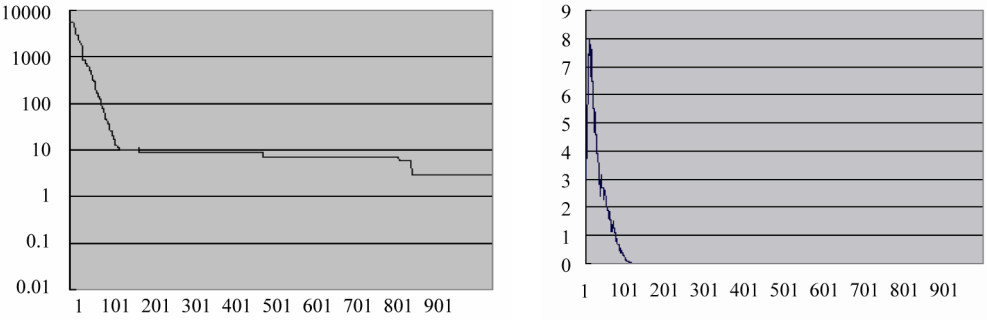
（a）　　（b）

图 3-7　f_4：Schwefel问题 2.21



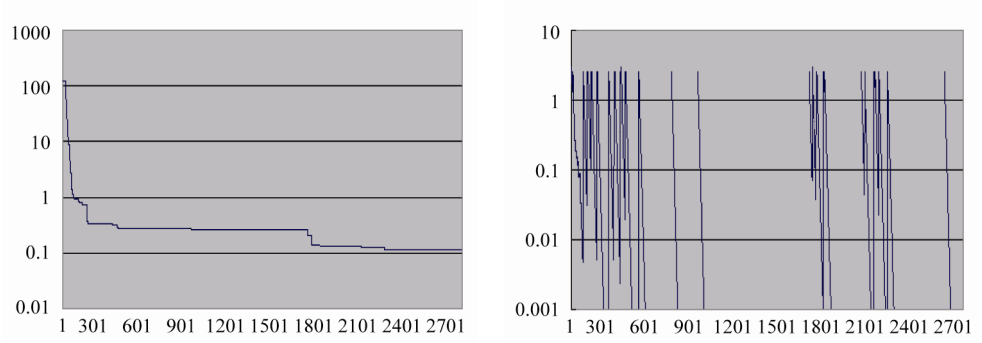
（a）　　（b）

图 3-8　f_5：广义 Rosenbrock函数



（a）　　（b）

图 3-9　f_6：阶梯函数



（a） （b）

图 3-10 f_7：带噪声的四次函数

文献[56]提供了 FEP 和 CEP 的相关数据，与之相比，采用 PHEP 方法 f_1～f_6 的收敛速度明显快于这两种方法，并且所得的最优解误差也小于 CEP 和 FEP。对于 f_7 数 PHEP 虽然能够快速收敛，最终结果略差于文献[56]提供的数据。考查变异向量变化曲线（图 3-4～图 3-10 中的(b)），对函数 f_1、f_2、f_4、f_6 变异向量经过放大之后，逐渐缩小，最后小于 10e-8；而对于 f_3、f_5 变异向量最终分别保持在 10e-6 和 10e-3 之间，这说明算法此时依然保持一定的变异成功率；对于 f_7，变异向量波动较大，这是由于 f_7 中有一个随机扰动的原因，由于 f_7 的搜索空间较小，这种扰动是不容忽视的，从而也造成了变异向量的扰动。从以上算法描述和相关曲线图中可以发现，对于 CEP 和 FEP，变异向量在式（3-1）的作用下，步长逐渐缩小，并且这种变化是盲目的。从这个角度，PHEP 对变异向量的调整属于自适应范畴，但这种自适应过程与传统的自适应策略不同（如 1/5 成功法则），PHEP 根据群体中个体分布情况来调整搜索步长，信息反映群体目前的状况，因此，相应地调整能够最有效地平衡算法的选择压力和群体压力，群体多样得到有效的保证，避免了算法的早熟。表 3-2 是采用不同方法 f_1～ f_7 所得最优解和迭代次数比较，其中，FEP 和 CEP 的数据是文献[56]提供的。

表 3-2 PHEP、FEP、CEP 方法试验结果比较（单峰函数）

No.	最优解	PHEP		FEP		CEP	
		t	最优解	t	最优解	t	最优解
f_1	0	1000	0	1500	5.7e-4	1500	2.2e-4
f_2	0	1000	0	2000	0.0081	2000	0.0026
f_3	0	1000	2.7280e-5	5000	0.016	5000	0.05

续表

No.	最优解	PHEP		FEP		CEP	
		t	最优解	*t*	最优解	*t*	最优解
f_4	0	5000	0.2435	5000	0.3	5000	6.17
f_5	0	10000	2.7757	20000	5.06	20000	13.61
f_6	0	1000	0	1500	0	1500	577.76
f_7	0	3000	0.1254	3000	7.6e-3	3000	1.8e-3

f_8～f_{13}是多峰函数，存在多个局部最优，表 3-3 所示为 PHEP、FEP 和 CEP 三种方法所获最优解和迭代次数的比较，PHEP 方法搜索到了函数 f_{10}、f_{12}、f_{13}全局最优，其他两个函数的实验结果也明显优于另外两种方法。

表 3-3　PHEP、FEP、CEP 方法试验结果比较（多峰函数）

No.	最优解	PHEP		FEP		CEP	
		t	最优解	*t*	最优解	*t*	最优解
f_8	-12569.5	9000	-12549.3	9000	-12554.5	9000	-7917.1
f_9	0	5000	0.079	5000	0.046	5000	89.0
f_{10}	0	1000	0	1500	0.018	1500	9.2
f_{11}	0	1000	3.7e-7	2000	0.016	2000	0.086
f_{12}	0	1000	0	1500	9.2e-6	1500	1.76
f_{13}	0	1000	0	1500	7.3e-5	1500	1.4

如何克服算法过早收敛到局部最优，一直是用演化方法求解优化问题的难点。本书提出的群体启发进化规划方法在这一点上较先前的一些方法有很大的提高，从 13 个测试结果上看，无论对于单峰函数还是多峰函数，PHEP 的结果和收敛速度都明显优于文献[56]中的方法。试验结果验证了 PHEP 方法具有较强的优化能力，为了进一步验证群体启发进化规划方法的性能，下一节将把该方法应用于求解更加复杂的高维优化问题。

3.5 用群体启发进化规划求解高维优化问题

3.5.1 高维优化

求解最优化问题过程中，若维度 n 的值相对较大（如 $n \geqslant 100$），则称之为高维优化问题。高维优化有一定的难度，对优化算法也是一个考验。本节将考查 PHEP 方法在高维情况下的性能。

之前讨论了个体 x 撞入某一最优 x_i^* 邻域的概率，在高维条件下，优化问题的欺骗性往往会变得更强，下面给出相关概念和分析。

假设 f 在搜索空间 S 上只存在有限多个最优解（包括全局最优解和局部最优解）。不妨设有 M 个全局最优解，且满足上文中的 3 个假设。用 S_{opt} 表示这 M 个点的集合，即 $S_{\text{opt}} = \{x_1^*, \mathbf{L}, x_M^*\}$，其中，$x_i^* = [x_{i1}^*, x_{i2}^*, \mathbf{L}, x_{in}^*]^{\text{T}}$，则必有 $\exists d > 0$，使得：

（1）$[x_i^* - d, x_i^* + d] \mathbf{I} [x_j^* - d, x_j^* + d] = \varnothing$，对 $\forall x_i^*, x_j^* \in S_{\text{opt}}$，$i \neq j$；

（2）令 $S_{\text{opt}\_d} = \mathop{\mathbf{U}}_{i=1}^{M} \left[x_i^* - d, x_i^* + d \right]$，若 $f(y) < f(x_d)$，其中，$y \in S$，$x_d \in S_{\text{opt}\_d}$，则必有 $y \in S_{\text{opt}\_d}$；

（3）对于所有的 $d' < d$，（1）、（2）都成立。

定义 3-1　以精度 d 接近：设 x 是 S 上一点，d 满足上述条件（1）～（3），若 $x \in S_{\text{opt}\_d} = \mathbf{U}[x_i^* - d, x_i^* + d]$，则称 x 以精度 d 接近全局最优。设 POP 是 S 上由多个个体组成的群体，如果 POP 中至少存在一个属于 S_{opt_d} 的个体，则称群体 POP 以精度 d 接近全局最优。特别地，若 $x \in [x_i^* - d, x_i^* + d]$（POP 中至少存在一个属于 $[x_i^* - d, x_i^* + d]$ 的个体），则称 x（群体 POP）以精度 d 接近全局最优 x_i^*。

定义 3-2　以精度 d 绝对接近：在定义 3-1 的基础上，如果每一个区间 $[x_i^* - d, x_i^* + d]$ 只包含一个极值点 x_i^*，则称 x（群体 POP）以精度 d 绝对接近全局最优。特别地，若 $x \in [x_i^* - d, x_i^* + d]$（POP 中至少存在一个属于 $[x_i^* - d, x_i^* + d]$ 的个体），则称 x（群体 POP）以精度 d 接近全局最优 x_i^*。

显然，上面的叙述并没有考虑 $x_i^* \in S_{\text{opt}}$ 是 S 的边界点时的情况，事实上，若某个 x_i^* 为 S 的边界点时，只须对包含 x_i^* 区间稍加修改即可，因此，在这里没有特别加以区分。

与低维优化问题相比，运用 EP 方法解决高维优化问题的复杂程度变得更高。

首先，考虑 S 中个体 $\forall x$ 变异后以精度 d 绝对接近全局最优 x_i^* 的概率 $P\{\forall x \xrightarrow{\text{Mutation}} x_d \mid x_d \in [x_i^* - d, x_i^* + d], \forall x \in S - S_{\text{opt}\_d}\}$，由于 n 维空间各维度之间相互独立，故有：$P\{\forall x \xrightarrow{\text{Mutation}} x_d \mid x_d \in \left[x_i^* - d, x_i^* + d\right], \forall x \in S - S_{\text{opt}\_d}\} = \prod_{j=1}^{n} P_i\left\{\forall x_i \to [x_{ij}^* - d_j, x_{ij}^* + d_j]\right\}$，其中，$P_i\left\{\forall x_i \to [x_{ij}^* - d_j, x_{ij}^* + d_j]\right\} = \dfrac{1}{\sqrt{2\pi}s} \int_{x_{ij}^* - d_j}^{x_{ij}^* + d_j} \exp\left(-\dfrac{t^2}{2s^2}\right) \mathrm{d}t$，由于 $P_i\left\{\forall x_i \to [x_{ij}^* - d_j, x_{ij}^* + d_j]\right\} < 1$，故而当问题的维数 n 增大时，个体 $\forall x$ 变异后以精度 d 绝对接近全局最优 x_i^* 的概率 $P\{\forall x \xrightarrow{\text{Mutation}} x_d \mid x_d \in [x_i^* - d, x_i^* + d], \forall x \in S - S_{\text{opt}\_d}\}$ 随着 n 的增大而减小。

其次，高维条件下，EP 的搜索环境受到很大影响，尤其对多峰函数的影响更加明显，而传统的 EP 方法通常使用函数 $f(\cdot)$来对每个个体进行评价，函数值小的个体存活的概率总是相对较大，这样，在“恶劣”的搜索环境下，由于搜索空间变得更加复杂，适应度相对较差的个体即使存活下来，在下一代中也很难通过变异产生出适应度较大的个体。也就是说，在以往 EP 方法的选择机制下，问题维度的增大，造成搜索难度得增大，同时也增大了适应度较大个体的存活概率，造成群体的多样性被破坏，加速了群体的过早收敛。文献[64]通过试验证实了 CEP 和 FEP 对高维优化的不适应。PHEP 方法放弃了传统的选择方式，在复杂搜索条件下，依然能够较好地保持群体的多样性。因此，问题维数的变化对算法影响相对较小。

3.5.2　实验结果

文献[57]提出了 MEP（Mixed Evolutionary Programming），MEP 同时使用了 Cauchy 分布函数和 Gaussan 分布函数，并按不同的概率使用 Cauchy 分布函数和 Gaussan 分布进行变异操作；文献[64]对 FEP 方法进行了一些改进，提出了 FEPCC（Fast Evolutionary Programming with Cooperative Coevolutionary），FEPCC 把整个系统分成多个模块，分别应用 FEP 于不同的模块。从得到的结果来看，在上述方法中，FEPCC 更适合解决高维

优化问题[64]。本书提出的群体启发进化规划（PHEP）在执行过程中，能够掌握群体中个体的分布情况，并根据这一情况来调整下一步变异向量的大小。目的是在算法的搜索过程中，尽可能地保持群体的多样性。与文献[64]中提供的 FEPCC 结果相比，在高维优化条件下，PHEP 方法有着更加出色的表现，更能适应高维条件下“恶劣”的搜索环境，搜索到的结果好于 FEPCC 方法所得到的结果。

为了方便比较，采用 f_1、f_2、f_4、f_6、f_8、f_9、f_{10}、f_{11} 作为测试函数，其中单峰函数（前 4 个函数）和多峰函数各占一半，有一定的代表性。设置 PHEP 参数，令 $m=100$、$l=0.3$、$t_l=1.05$、$g=0.8$、$t_g=1.15$、$a=0.65$、$t_a=0.85$、$n=0.3$、$b=0.45$、$t_b=1.05$，PHEP 运行 50 次，所得最优解平均值与文献[64]提供的 FEPCC 比较如表 3-4 和表 3-5 所示。

表 3-4　高维条件下 PHEP、FEPCC 方法试验结果比较（单峰函数）

No.	维数	FEPCC	PHEP
f_1	100	6.8e-9	0.0
	250	2.1e-8	0.0
	500	4.9e-8	0.0
	750	3.4e-8	0.0
f_2	100	2.1e-4	8.728e-6
	250	4.3e-4	3.424e-5
	500	1.3e-3	5.878e-5
	750	2.1e-3	7.767e-4
f_4	100	3.8e-5	6.509e-6
	250	5.8e-5	1.646e-6
	500	0.0	3.989e-5
	750	0.0	2.546e-5
f_6	100	0.0	0.0
	250	0.0	0.0
	500	0.0	0.0
	750	0.0	0.0

表 3-5　高维条件下 PHEP、FEPCC 方法试验结果比较（多峰函数）

No.	维数	FEPCC	PHEP
f_8	100	-41867.3	-38745.6
	250	-1046772	-94792.4
	500	-209316.4	-204566.3
	750	-313995.8	-309987.2
f_9	100	2.6e-2	5.119e-4
	250	4.8e-2	3.645e-3
	500	1.4e-1	3.333e-3
	750	1.6e-1	7.346e-2
f_{10}	100	1.7e-4	2.106e-4
	250	3.5e-4	3.295e-4
	500	5.7e-4	2.723e-4
	750	7.8e-4	4.322e-4
f_{11}	100	4.7e-2	7.904e-3
	250	2.5e-2	1.426e-2
	500	2.9e-2	1.988e-2
	750	6.1e-2	3.756e-2

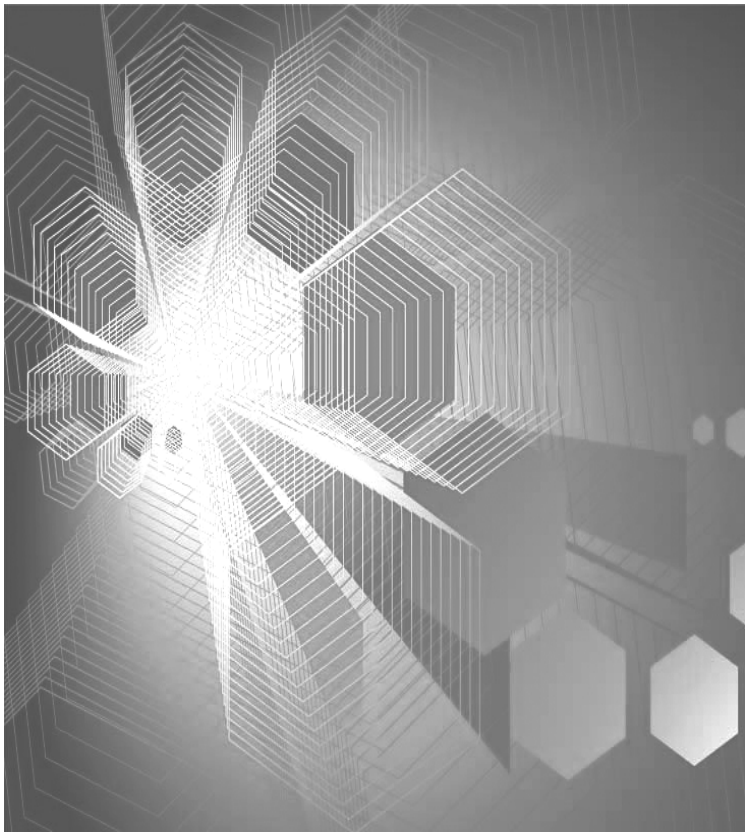
从表 3-4 和表 3-5 中可以看出，即使在高维条件下，采用 PHEP 方法也可以搜索到函数 f_1、f_4 的全局最优解，f_1 相对简单，但当搜索空间维度较大时，搜索到全局最优，依然有较大难度；f_4 是阶梯函数，每一个“台阶”都有相同的评价函数值，这会给个体选择造成一定的难度；其他函数实验结果明显优于文献[64]中提供的结果，并保持了较好的搜索效率。从表 3-5 中可以看出，在高维条件下，PHEP 方法获得了较好的搜索结果，这应该归功于算法的变异调整策略，针对不同的群体有不同变异操作，保证了群体的多样性，一定程度上克服了算法“早熟”现象的发生；同时，使用参数 $O_{\text{direct\_sel}}$、O_{normal}、O_{mut}、$O_{\text{sel\_mut}}$ 来把握群体中个体的分布情况，根据个体的分布调整变异步长 h_i，使之在恰当的时候增大或减小，较大程度地避免了算法在步长调整上的盲目性，使 PHEP 方法在高维搜索环境下，有更好的适应性。

3.6 小结

本章对经典进化规划、群体启发进化规划及相关知识进行了阐述，并对不同的变异分布进行了分析，给出了个体基于 Gauss 变异搜索到全局最优的概率估计。用进化规划方法求解优化问题的关键是如何克服算法“早熟”现象的发生，算法发生“早熟”意味着无法搜索更好的个体（潜在解），这时，群体中的个体分布集中，多样性差。群体只有产生更新的个体，开辟新的搜索区域才能摆脱这种困境。群体启发进化规划就是根据个体的分布情况调整变异向量。同时，群体中的 4 个子集都保持各自的变异策略，群体多样性得到了较好的保持。本质上，PHEP 对变异向量的调整属于自适应范畴，但这种自适应过程与传统的自适应策略不同，PHEP 根据群体中个体分布情况来调整搜索步长，这种分布信息反映了群体目前的状况，因此，相应的调整能够最有效地平衡算法的选择压力和群体压力。

与其他求解优化问题的演化计算方法一样，测试函数是评价算法性能的“测试床”。目前，只能通过测试来比较不同算法的优劣和定性地对算法加以分析，本章采用有代表性的 Benchmark 函数对 PHEP 进行了测试和比较，并将之应用于求解高维优化问题，事实证明，PHEP 改变了以往 EP 的选择机制，在进化过程中较好地保持了群体的多样性。较从前一些 EP 方法更能适应高维条件下“恶劣”的搜索环境，有较好的应用价值。

第 4 章　粒子群优化



粒子群优化（PSO）方法是由 Kennedy 和 Eberhart 于 1995 年提出的一种进化计算技术[65][66]，广泛应用于各类优化问题，其基本思想源于对鸟群捕食行为的研究。鸟群捕食的基本原理可以简单概括为：在鸟群中，每个个体都承担着搜索食物的任务，同时每个个体又分享着来自其他个体的信息，这样每只鸟下一步将要搜索的区域受到自身搜索经验和其他个体搜索经验两个因素的影响，个体根据这两个因素来确定下一步最有希望找到食物的位置。目前，有很多关于 PSO 方法的研究，其中文献[67]提出了一种自由参数 PSO 方法，以解决 PSO 方法参数选择较难的问题；文献[68]提出了参数自适应粒子群优化方法，许多关于 PSO 的改进方法都是基于这种参数自适应的思想；文献[69]、文献[70]把 PSO 方法成功地应用于离散搜索空间；文献[71]介绍了一种控制粒子距离的方法，通过控制粒子间的距离来保证粒子的多样性。这些方法广泛应用于函数优化、神经网络训练、模式分类、模糊系统控制等各种领域[72]~[75]。本章对 PSO 方法进行阐述和分析，讨论参数设置与粒子收敛性之间关系的证明，介绍了两种 PSO 方法的改进方法：模拟退火粒子群优化（Particle Swarm Optimization with Simulated Annealing，PSOwSA）和有分工策略的粒子群优化（Particle Swarm Optimization with Division Of Work，PSOwDOW）。在 PSOwSA 中，当粒子下一步位置比当前位置好时，粒子移动到下一步位置；反之，若下一个位置比当前位置差，则粒子并不直接移动到下一个位置，而是以某一概率进行移动，且通过温度 T 来控制这一概率。若把标准 PSO 看成对鸟类觅食等群体行为的简单模拟，则在 PSOwSA 中的“鸟”更加聪明，它不是盲目地直接扑向下一个位置，而是以某种概率“试探”后再行动。因此，当温度下降得足够慢时，粒子不会轻易地跳出有“希望”的搜索区域，从而增强了粒子的局部搜索能力。在 PSOwDOW 中，为增强群体的局部搜索能力，把粒子群划分为 3 个互不交叉的子群体并作分工，在执行过程中这 3 个群体不断地发生变化。这种处理方式较好地平衡了局部“搜索”并“开拓”了新领域之间的关系，在计算量没有明显增加的前提下，计算结果却有较大的提高。下面就对上述内容加以详细阐述。

4.1　标准粒子群优化方法

在解决优化问题上，每个优化问题的潜在解都是搜索空间中的一个“粒子”。每个粒子 i 可以用三元组 (x_i, v_i, p_i) 表示，其中，x_i 表示粒子当前的位置，v_i 表示粒子当前的速度，p_i 表示粒子本身搜索过的最好的位置。有时为了描述上的简便，用粒子的位置 x_i 表示粒子本身，所有粒子都有一个由评价函数（被优化函数）f 决定的适应度值（Fitness Value），粒子当前的速度 v_i 决定了它们的“飞行”方向和“飞行”距离。此后一直假设问题的搜索空间是 D 维空间上的一个超立方体，用 S 表示，且 $S=[a_1,b_1]\times\mathbf{L}\times[a_D,b_D]$。在每一次迭代中，粒子 x 按式（4-1）进行更新：

$$\begin{aligned} v_{t+1} &= c\left(wx_t + c_1r_1(x_t - p_i) + c_2r_2(x_t - p_g)\right) \\ x_{t+1} &= x_t + v_{t+1} \end{aligned} \qquad (4\text{-}1)$$

其中，p_i 为粒子 i 经过的最好的位置（自身经验），p_g 表示的是群体中所有粒子经过的最好位置（群体经验）；c_1、c_2 为正常数，r_1、r_2 为[0,1]上的随机数；w 为惯性参数；c 为压缩因子。标准 PSO 方法如算法 4-1 所述。在算法中，当粒子在某一维度上飞出问题的搜索范围时，令它在这一维度上等于搜索区间的上界或者下界。

算法 4-1　粒子群优化算法

0	Algorithm：Particle Swarm Optimization, PSO
1	Initialize Population
2	Do
2.1	For i = 1 to Population Size
2.1.1	If $f(x_i) < f(p_i)$ then $p_i = x_i$，$p_g = \min(p_i)$
2.1.2	For d = 1 to Dimension
2.1.2.1	$v_{id} = v_{id} + c_1j_1(p_{id} - x_{id}) + c_2j_2(p_{gd} - x_{id})$
2.1.2.2	$x_{id} = x_{id} + v_{id}$
2.1.2.3	if $x_{id} < a_d$ then $x_{id} = a_d$，if $x_{id} > b_d$ then $x_{id} = b_d$
2.1.3	End for
2.2	End for
3	Until termination criterion is met

J. Kennedy 从社会心理学的角度对粒子的行为进行了解释。在鸟类或鱼类觅食的过程中，每个个体的行动都遵循两个简单的规则，一个是依据个体的经验进行搜索，即每个个体都趋向于它曾经搜索过的最好位置；另一个是依据群体的经验进行搜索，即每个个体都参考其相邻个体的行为，这使得找到较好位置的个体会吸引其他个体到它的附近进行搜索。遵循上述两个简单的规则，觅食的鸟类或鱼类群体有时可以绵延数十平方千米。每个个体的简单行为可以产生整个群体的复杂的搜索，这就是粒子群优化算法的群体智能原理。

式（4-1）中的 $c_1r_1(x_t - p_i)$ 项表示个体的经验或个体对环境的认知，$c_2r_2(x_t - p_g)$ 项表示群体的经验或群体对环境的认知。在一个群体中，个体的行为不仅受个体经验的影响，还受群体认知的左右。因此，式（4-1）考虑了两方面的因素，两个因素对个体行为影响的大小通过加速系数 c_1、c_2 调节。若粒子的行为仅考虑个体经验，那么式（4-1）就变成 $wx_t + c_1r_1(x_t - p_i)$；若粒子的行为仅考虑群体经验，那么式（4-1）就变成 $v_{t+1} = wx_t + c_2r_2(x_t - p_g)$。$wv_{ij}(t)$ 项表示前一次迭代中速度的惯性值。

许多学者也尝试把变异、交叉和选择算子显式地引入粒子群优化算法。Angeline 在 1999 年提出了具有选择操作的粒子群优化算法[81]，M. Løvbjerg 等在 2001 年提出了具有交叉操作的粒子群优化算法[82]等，这些算法在某些方面或针对某些具体的优化问题都优于基本的粒子群优化算法。

粒子群优化算法可分为如下两种模型。

1．Gbest 模型

定义 $p_g(t)$ 为整个群体的迄今为止的历史最好位置，简称全局最优粒子。粒子的位置和速度按式（4-1）进行更新。Gbest 模型的特点是整个群体维持一个全局最优粒子，全局最优粒子吸引其他粒子趋向于它所在的位置。因此，如果全局最优粒子不能及时更新，会造成算法的早熟收敛（Premature Convergence）。

2．Lbest 模型

Lbest 模型对种群内各粒子分别定义一个局部最优粒子，这就使得种群内存在多个吸引点，从而可以避免 Gbest 模型的早熟收敛问题。假设粒子的索引取模 s 的余数，定义粒子 i ，i=1,2,$\mathbf{L}$, s 的大小为 l 的邻域为：$N_i=\{\bar{y}_{i-l}(t),\bar{y}_{i-l+1}(t),\mathbf{L},\bar{y}_{i-1}(t),\bar{y}_i(t),\bar{y}_{i+1}(t),\mathbf{L},\bar{y}_{i+l}(t)\}$，取 $\bar{\bar{y}}_i(t)\in N_i$，满足 $f(\bar{\bar{y}}_i(t))=\min\{f(a)\,|\,a\in N_i\}$。粒子在每个邻域内都按照式（4-1）进行修正。

需要特别说明的是，N_i 中的粒子彼此没有任何关系，仅是索引顺序的相邻，因此，N_i 的计算并不需要聚类（Clustering）操作。

显然，当取 l=s 时，Lbest 模型就变成 Gbest 模型，因此，Gbest 模型是 Lbest 模型的特殊情况。取 l=1 的 Lbest 模型不容易陷入局部极小点，但与 Gbest 模型相比，算法收敛速度很慢[42]。Lbest 模型可以通过在算法中设置多个 Gbest 模型的种群模拟实现。

4.2　二进制粒子群优化算法

如果要解决的优化问题是二进制优化问题，即搜索的空间是由定长的二进制比特串构成，那么，使用二进制优化算法可能更合适。或更一般地，解决离散的优化问题更适合采用离散的优化算法。1997 年，J. Kennedy 和 R. C. Eberhart 提出了一种二进制粒子群优化（Binary Particle Swarm Optimization，BPSO）算法，在这个模型中，个体进行二进制决策的概率是一个与个体和群体相关的函数，定义如下：

$$P(x_{id}(t)=1)=f(x_{id}(t-1),v_{id}(t-1),p_{id},p_{gd}) \qquad (4\text{-}2)$$

其中，$P(x_{id}(t)=1)$ 是个体 i 为位串上第 d 位选择 1 的概率，当然选择零的概率为 $1-P$ 。

$x_{id}(t)$ 是个体 i 的位串位置 d 的当前状态。

t 是当前的时间步，而 $t-1$ 是前一步。

$v_{id}(t)$ 代表个体作一个或另一个选择的倾向，它将决定一定概率阈值，并运用 S 形函数将其限制于需要的范围，并且有时它适于用作概率阈值的性质。

p_{id} 是至今为止发现的最好状态，例如，如果 x_{id} 为 1 时个体取得最大的成功，那么 p_{id} 为 1；若 x_{id} 为 0 时个体取得最大的成功，那么 p_{id} 为 0。

p_{gd} 是邻域的最好状态，如果邻域的任意成员在处于1状态时取得最大的成功，则 p_{gd} 为 1；否则为 0。

在二进制空间中，粒子的移动是通过翻转位值实现的，而粒子的速度描述了每次迭代改变的位数，或某粒子在 t 和 $t-1$ 时刻的取值之间的海明距离。为了与连续空间粒子群算法在表述上保持一致性，忽略其中的时刻表示，并且其中参数的含义保持不变。粒子群算法的离散二进制模型的速度和位置更新等式为

$$v_{id} = v_{id} + c_1 \mathrm{rand}_1(\,)(p_{id} - x_{id}) + c_2 \mathrm{rand}_2(\,)(p_{gd} - x_{id})$$

$$\text{if } \mathrm{rand}(\,) < S(v_{id}) \quad \text{then } x_{id} = 1 \quad \text{else } x_{id} = 0 \tag{4-3}$$

其中，$S(v_{id}) = 1/(1+\exp(-v_{id}))$ 为 Sigmoid 函数，rand( ) 为[0，l]上的随机数。速度分量 v_{id} 决定了位置分量 x_{id} 取 1 或 0 的概率，v_{id} 越大，则 x_{id} 取 1 的概率越大。在离散二进制模型中，仍保留了 $V_{\max}$，它起限制 x_{id} 取 1 或 0 的最终概率的作用。

假设搜索空间 S 是 d 维的，目标函数 $f(\boldsymbol{X})$ 为最小化函数。在二进制粒子群算法中，第 i 个粒子表示为有序三元组 $(\boldsymbol{X}_i, \boldsymbol{V}_i, \boldsymbol{P}_i)$，其中，$\boldsymbol{X}_i = (x_{i1}, x_{i2}, \mathbf{L}, x_{id})^{\mathrm{T}}$ 表示粒子当前位置，$x_{ij} \in \{0,1\}$，$j = 1,2,\mathbf{L},n$；$\boldsymbol{V}_i = (v_{i1}, v_{i2}, \mathbf{L}, v_{id})^{\mathrm{T}}$ 表示当前速度，$\boldsymbol{V}_i \in S$；$\boldsymbol{P}_i = (p_{i1}, p_{i2}, \mathbf{L}, p_{id})^{\mathrm{T}}$ 表示粒子已搜索过的个体最好位置（个体经验），其中，$p_{ij} \in \{0,1\}$，$j = 1,2,\mathbf{L},d$。

又设种群由 s 个粒子构成，且所有粒子经过的全局最好位置（集体经验）记为 $\boldsymbol{P}_g = (p_{g1}, p_{g2}, \mathbf{L}, p_{gd})^{\mathrm{T}}$，$p_{gj} \in \{0,1\}$，$j = 1,2,\mathbf{L},d$，则粒子 $(\boldsymbol{X}_i, \boldsymbol{V}_i, \boldsymbol{P}_i)$ 的速度与位置的演化公式为

$$v_{ij}(t+1) = v_{ij}(t) + c_1 r_1 (p_{ij}(t) - x_{ij}(t)) + c_2 r_2 (p_{gj}(t) - x_{ij}(t)) \tag{4-4}$$

$$x_{ij}(t+1) = \begin{cases} 0, & \mathrm{sign}(x_{ij}(t+1)) \leqslant r_3 \\ 1, & \mathrm{sign}(x_{ij}(t+1)) > r_3 \end{cases} \tag{4-5}$$

其中，$1 \leqslant i \leqslant s$，$1 \leqslant j \leqslant d$，sign($x$) 是一个模糊函数且sign($x$)=1/(1+e^{-x})；$r_1$、$r_2$、$r_3$ 是随机数且$r_1, r_2, r_3 \in [0,1]$。此外，若 $f(\boldsymbol{X}_i(t+1)) \geqslant f(\boldsymbol{P}_i(t))$，则 $\boldsymbol{P}_i(t+1) = \boldsymbol{P}_i(t)$；若 $f(\boldsymbol{X}_i(t+1)) < f(\boldsymbol{P}_i(t))$，则 $\boldsymbol{P}_i(t+1) = \boldsymbol{X}_i(t+1)$，$\boldsymbol{P}_g(t) \in \{P_1(t), P_2(t), \mathbf{L}, P_s(t)\}$，且 $f(P_g(t)) = \min\{f(P_1(t)), f(P_2(t)), \mathbf{L}, f(P_s(t))\}$。为了讨论方便，将式（4-4）改写为

$$v_i(t+1) = v_i(t) + c_1 r_1 (p_i(t) - x_i(t)) + c_2 r_2 (p_g(t) - x_i(t))$$

其中，$1 \leqslant i \leqslant s$，如果令 $\boldsymbol{G}_1 = c_1 r_1 (p_i(t) - x_i(t))$，$\boldsymbol{G}_2 = c_2 r_2 (p_g(t) - x_i(t))$，则可得

$$v_i(t+1) = v_i(t) + \boldsymbol{G}_1 + \boldsymbol{G}_2, \quad i = 1, 2, \mathbf{L}, s \tag{4-6}$$

可以看出第 i 个粒子 $(\boldsymbol{X}_i, \boldsymbol{V}_i, \boldsymbol{P}_i)$ 的速度 $\boldsymbol{V}_i$ 包含三部分内容：第一部分 $v_i(t)$ 是第 t 代粒子的飞行速度，第二部分 $\boldsymbol{G}_1$ 和第三部分 $\boldsymbol{G}_2$ 分别是对 $v_i(t)$ 的修正，其中，$\boldsymbol{G}_1$ 是该粒子前 t 代的个体最好位置 $p_i(t)$ 对此修正的影响程度，$\boldsymbol{G}_2$ 是整个群体前 t 代的全局最好位置 $p_g(t)$ 对修正的影响程度。如果 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 中含有 0 的分量较多，则 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 对粒子速度的修正程度将减小，粒子的飞行状态不能得到充分的调整，使得种群多样性和全局收敛性大大降低，影响了算法的性能。设 $\boldsymbol{X} = (x_1, x_2, \mathbf{L}, x_n)^{\mathrm{T}}$ 是一个 n 维二进制向量，令 $\| \boldsymbol{X} \|$ 表示向量 $\boldsymbol{X}$ 中值为 0 的分量 $\boldsymbol{X}_j$ 的个数（$1 \leqslant j \leqslant n$）。又设 $\boldsymbol{G}_1 = (g_{11}, g_{12}, \mathbf{L}, g_{1d})^{\mathrm{T}}$，$\boldsymbol{G}_2 = (g_{21}, g_{22}, \mathbf{L}, g_{2d})^{\mathrm{T}}$，则 $g_{1j} = c_1 r_1 (p_{ij}(t) - x_{ij}(t))$，$g_{2j} = c_2 r_2 (p_{gj}(t) - x_{ij}(t))$，$1 \leqslant j \leqslant d$。在等概率情况下，由于 $p_{ij}(t)$，$x_{ij}(t) \in \{0, 1\}$，则 $(p_{ij}(t) - x_{ij}(t)) = 0$ 的概率为 0.5，于是 g_{1j} 值为 0 和不为 0 的概率均是 0.5。这样在 $\boldsymbol{G}_1$ 的每一次计算中，其每个分量 g_{1j}（$1 \leqslant j \leqslant d$）的值等于 0 或不等于 0 的概率均是 0.5。由于各分量 g_{1j} 的取值是相互独立的，每一次计算是一次独立的 Bemoulli 实验，那么对 $\boldsymbol{G}_1$ 的一次计算由其 d 个分量的 d 次计算构成了 d 重 Bemoulli 实验，根据二项概率公式可得 $\| \boldsymbol{G}_1 \| \geqslant d/2$ 的概率：

$$\text{prob}\ [\| \boldsymbol{G}_1 \| \geqslant d/2] = \sum_{k=\lfloor d/2 \rfloor + 1}^{d} C_d^k (1/2)^k (1/2)^{d-k} / (1/2)^d$$

所以，$2 \sum_{k=\lfloor d/2 \rfloor + 1}^{d} C_d^k (1/2)^k (1/2)^{d-k} \geqslant \sum_{k=1}^{d} C_d^k (1/2)^k (1/2)^{d-k} = 2^d$，于是由上式得 prob[$\| \boldsymbol{G}_1 \| \geqslant d/2$] $\geqslant (2^d/2)/2^d = 0.5$，这表明在二进制粒子群算法中，$\boldsymbol{G}_1$ 中各个分量 g_{1j}（$1 \leqslant j \leqslant d$）

等于 0 的概率不小于 1/2；同样的道理，$\boldsymbol{G}_2$ 的各个分量 g_{2j} 等于 0 的概率也不小于 1/2。注意结论与 t 的大小无关，因此，二进制粒子群算法从第 1 代开始 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 中值为 0 的分量个数不少于一半，而 $\boldsymbol{G}_1$ 和 $\boldsymbol{G}_2$ 中这些值等于 0 的分量对粒子 $(\boldsymbol{X}_i,\boldsymbol{V}_i,\boldsymbol{P}_i)$ 的速度不会产生任何影响，也就是说，在这些分量的方向上粒子速度不会得到改变与修正，这样会使种群的多样性必然降低，削弱了算法的全局搜索能力，搜索到问题最优解的机会大大减少，而且这种情形并不会随着种群演化代数的不断递增而得到改变。

粒子群算法（PSO）与遗传算法（GA）都是进化计算方法。从算法的步骤来看，两者基本类似，都遵循以下过程：

① 群体随机初始化；

② 计算群体的每一个体适应值，通常适应值与最优解关联；

③ 群体按个体适应值概率进行复制；

④ 判断终止条件。如果满足终止条件，算法停止；否则，转至步骤②。

同时，算法的基本点都是基于适应度的概率计算。两种算法的主要区别如下。

首先，在进化算子上，遗传算法采用遗传操作算子如交叉（Crossover）和变异（Mutation），粒子群算法是通过对速度的修改来完成进化和搜索的。从某种意义上讲，PSO 的搜索比 GA 更具随机性，更不容易落入局部最优。

其次，与遗传算法比较，粒子群算法的信息共享机制不同，在遗传算法中，整个种群的移动是比较均匀地向最优区域移动；在粒子群算法中，信息的流动是单向的，整个搜索过程跟随当前最优解的过程。与遗传算法比较，在大多数情况下，所有的粒子可能更快地收敛于最优解。此外，粒子群算法基本不受问题峰数增加的影响，受问题维数的影响也很小。

最初的 PSO 是从解决连续优化问题发展起来的，在大多数测试函数中，二进制 PSO 比遗传算法速度快，尤其在问题的维数增加时。

二进制粒子群优化算法的速度迭代公式与连续粒子群优化算法相同，但加速系数 w，以及 c_1 和 c_2 的意义与其在连续粒子群优化算法中的意义完全不同。一些文献规定粒子

的最大速度 v_{max} 满足$|v_{max}|=4$，这使得 $0.018 \leqslant \mathrm{Sig}(v_{ij}(t+1)) \leqslant 0.982$，类似于遗传算法中的变异操作概率 P_m，它避免算法的早熟收敛。v_{max} 越大，算法越倾向于当前最优解附近的局部搜索；v_{max} 越小，算法倾向于全局搜索。特别地，当 $v_{max}=0$ 时，算法变成纯随机搜索算法[52]；目前，对加速系数 w、c_1 和 c_2 的研究成果不多，一般取 $w=c_1=c_2=1$。

任取粒子群中的一个粒子 i，分以下几种情形讨论当迭代次数 $t \to +\infty$ 时粒子的运动轨迹。

1．当 $x_{ij}^{(p)}(t)= x_j^{(g)}(t)$时

若 $x_{ij}(t) \neq x_{ij}^{(p)}(t)$和 $x_j^{(g)}(t)$，随迭代次数 t 增加；当 $x_{ij}^{(p)}(t)= x_j^{(g)}(t)=1$ 时，$v_{ij}(t) \to +\infty$，这使得 $x_{ij}(t) \to 1$；当 $x_{ij}^{(p)}(t)= x_j^{(g)}(t)=0$ 时，$v_{ij}(t) \to -\infty$，这使得 $x_{ij}(t) \to 0$。因此，粒子会收敛到全局最优粒子所在位置。加速系数 c_1 和 c_2 越大，粒子收敛的速度越快；如果加速系数 $w>1$，那么 w 越大，粒子收敛的速度越快。如果惯性权值系数 $w<1$，当达到 $x_{ij}(t)=x_j^{(g)}(t)$和 $x_{ij}^{(p)}(t)$，加速系数 c_1 和 c_2 不再起作用，随迭代次数 t 增加，w 将使 $v_{ij}(t) \to 0$，即使得 $x_{ij}(t)$产生变异的概率趋近于 0.5。因此，$w<1$ 会使 $x_{ij}(t) \to x_j^{(g)}(t)$和 $x_{ij}^{(p)}(t)$的过程中产生变异，且 w 越小，产生变异的概率就越大。图 4-1 和图 4-2 给出了两组典型参数取值下 $x_{ij}(t)$和 $v_{ij}(t)$的运动轨迹。

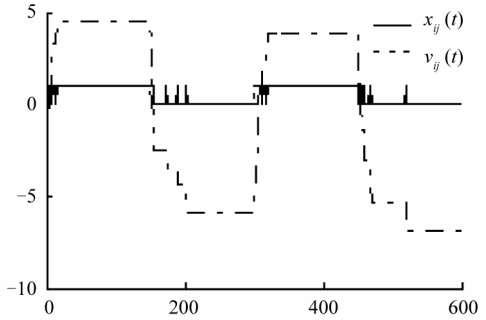


图 4-1　BPSO粒子的位置和速度变化（w=1.2, $c_1=c_2$=1）

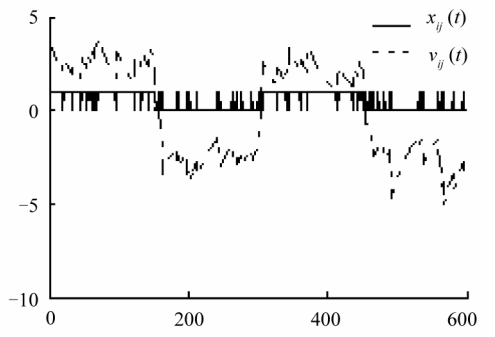


图 4-2　BPSO粒子的位置和速度变化（w=0.97, c_1=c_2=1）

图 4-1 和图 4-2 中取 $x_{ij}{}^{(p)}(t)$= $x_j{}^{(g)}(t)$=$\begin{cases}1, t=0\sim150, 301\sim450\\0, t=151\sim300, 451\sim600\end{cases}$，对粒子的最大速率 $v_{\max}$ 没有限制，初始速度 $v_{ij}(0)$取[-5,+5]的随机数，初始位置取 $x_{ij}(0)$=0。由图 4-1 可知，当 w>1 时，只要 $x_{ij}(t)\neq x_j{}^{(g)}(t)$和 $x_{ij}{}^{(p)}(t)$，随迭代次数 t 增加，$x_{ij}(t)\rightarrow x_j{}^{(g)}(t)$和 $x_{ij}{}^{(p)}(t)$，在开始阶段 $x_{ij}(t)$会存在一些变异，但每次变异，都使 $|v_{ij}(t)|$ 增大，进而使变异的概率变小；实验条件相同，仅取 w<1，由图 4-2 可知，在 $x_{ij}(t)\rightarrow x_j{}^{(g)}(t)$和 $x_{ij}{}^{(p)}(t)$的过程中，产生变异的概率有较大的增加。

2．当 $x_{ij}{}^{(p)}(t)\neq x_j{}^{(g)}(t)$时

当 $x_{ij}{}^{(p)}(t)\neq x_j{}^{(g)}(t)$时，必有 $x_{ij}(t)=x_{ij}{}^{(p)}(t)$或 $x_{ij}(t)=x_j{}^{(g)}(t)$，当 $x_{ij}(t)=x_{ij}{}^{(p)}(t)$时，加速系数 c_1 不起作用，c_2 起作用，随迭代次数 t 增加，$x_{ij}(t)\rightarrow x_j{}^{(g)}(t)$；而一旦 $x_{ij}(t)=x_j{}^{(g)}(t)$，加速系数 c_2 不起作用，c_1 起作用，那么随迭代次数 t 增加，$x_{ij}(t)\rightarrow x_{ij}{}^{(p)}(t)$。因此，粒子的位置坐标会在 $x_{ij}{}^{(p)}(t)$和 $x_j{}^{(g)}(t)$之间振荡。图 4-3 和图 4-4 给出了两组典型参数取值的粒子运动轨迹。

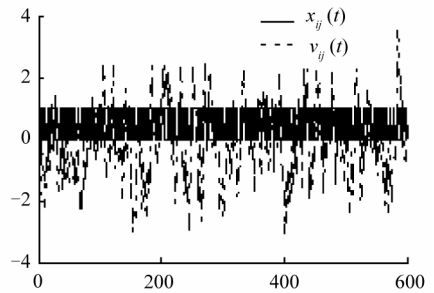


图 4-3　BPSO粒子的位置和速度变化（w=1, c_1=c_2=1.5）

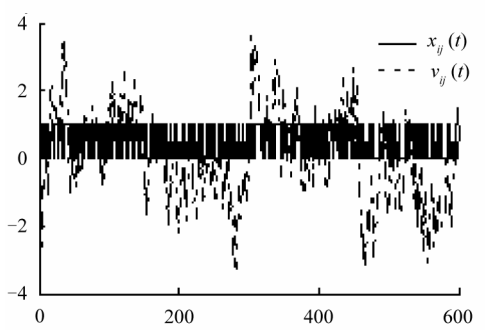


图 4-4　BPSO粒子的位置和速度变化（w=1, c_1=0.8, c_2=1.5）

图 4-3 和图 4-4 中取 $x_j^{(g)}(t)=\begin{cases}1, t=0\sim150, 301\sim450\\0, t=151\sim300, 451\sim600\end{cases}$，$x_{ij}^{(p)}(t)=1-x_j^{(g)}(t)$，对粒子的最大速率 $v_{\max}$ 没有限制，初始速度 $v_{ij}(0)$取[-5,+5]的随机数，初始位置取 $x_{ij}(0)=0$。当取 $c_1=c_2$（=1.5）时，无论 $x_j^{(g)}(t)$和 $x_{ij}^{(p)}(t)$的取值如何，$x_{ij}(t)$均在[0,1]均匀振荡，此时粒子群接近于随机搜索；当取 c_1=0.8，c_2=1.5（$c_2>c_1$）时，由于 c_2 的作用，尽管 $x_{ij}(t)$也在[0,1]振荡，但更倾向于 $x_{ij}(t)\rightarrow x_j^{(g)}(t)$，即粒子具有局部搜索特性。因此，$c_1$ 和 c_2 应适当选取。

3．限制 $v_{\max}$ 取值对粒子运动轨迹的影响

若对 $v_{\max}$ 做出约束，则随 $t\rightarrow+\infty$，$v_i(t)\rightarrow+v_{\max}$（或 $v_i(t)\rightarrow-v_{\max}$），可使得在 $x_{ij}(t)\rightarrow x_{ij}^{(p)}(t)$或 $x_j^{(g)}(t)$的过程中，存在一定概率的变异。图 4-5 给出了对 $v_{\max}$ 限制的粒子运动轨迹，实验条件与图 4-1 完全相同，仅限定 $v_{\max}$=2，与图 4-1 相比，粒子运动轨迹出现较大的振荡，限制 $v_{\max}$ 与取惯性权值系数 w<1 有相似的效果。

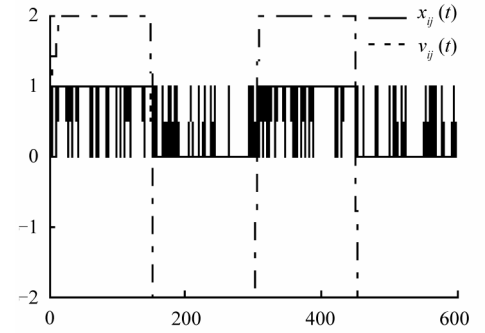


图 4-5　BPSO粒子的位置和速度变化（w=1.2, c_1=c_2=1, $v_{\max}$=2）

4.3 参数设置

定理 4-1 假设粒子 x_i 搜索到的最好位置 p_i 和整个群体搜索到的最好位置 p_g 固定不变，$f_1 = c_1 r_1$、$f_2 = c_2 r_2$ 为常数，压缩因子 $c = 1$，搜索空间没有限制，则当 $t \to \infty$ 时，粒子 x_i 最终将收敛到以 p_i 和 p_g 为端点的线段上。

证明：不妨设搜索空间的维数是 1，粒子 x 在 $t+1$ 步的位置由下式确定：

$$\begin{aligned} v_{t+1} &= wx_t + f_1(x_t - p_i) + f_2(x_t - p_g) \\ x_{t+1} &= x_t + v_{t+1} \end{aligned} \tag{4-7}$$

对式（4-7）改写并整理，得

$$x_{t+1} = (1 - f_1 - f_2)x_t + f_1 p_i + f_2 p_g + wv_t \tag{4-8}$$

把 $v_t = x_t - x_{t-1}$ 带入式（4-8）得

$$x_{t+1} = (1 + w - f_1 - f_2)x_t - wx_{t-1} + f_1 p_i + f_2 p_g \tag{4-9}$$

用矩阵和向量表示成

$$\begin{bmatrix} x_{t+1} \\ x_t \\ 1 \end{bmatrix} = \begin{bmatrix} 1 + w - f_1 - f_2 & -w & f_1 p_i + f_2 p_g \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ x_{t-1} \\ 1 \end{bmatrix} \tag{4-10}$$

式（4-10）中矩阵的特征方程为

$$(1 - l)\left[w - l(1 + w - f_1 - f_2) + l^2 \right] = 0 \tag{4-11}$$

求式（4-11）得到矩阵的 3 个特征值

$$\begin{aligned} l_1 &= 1 \\ l_2 &= \frac{1 + w - f_1 - f_2 + g}{2} \\ l_3 &= \frac{1 + w - f_1 - f_2 - g}{2} \end{aligned}$$

其中

$$g = \sqrt{(1 + w - f_1 - f_2)^2 - 4w}$$

令

$$\boldsymbol{A}=\begin{bmatrix}1+w-f_1-f_2 & -w & f_1p_i+f_2p_g\\ 1 & 0 & 0\\ 0 & 0 & 1\end{bmatrix},\quad \boldsymbol{\Lambda}=\begin{bmatrix}l_1 & 0 & 0\\ 0 & l_2 & 0\\ 0 & 0 & l_3\end{bmatrix}$$

则存在可逆矩阵 $\boldsymbol{Q}$，使 $\boldsymbol{A}=\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}$，因此，式（4-10）可以写成

$$\begin{bmatrix}x_{t+1}\\ x_t\\ 1\end{bmatrix}=\boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1}\begin{bmatrix}x_t\\ x_{t-1}\\ 1\end{bmatrix} \tag{4-12}$$

即

$$\begin{bmatrix}x_{t+1}\\ x_t\\ 1\end{bmatrix}=\boldsymbol{Q}\begin{bmatrix}1 & 0 & 0\\ 0 & l_2{}^t & 0\\ 0 & 0 & l_3{}^t\end{bmatrix}\boldsymbol{Q}^{-1}\begin{bmatrix}x_1\\ x_0\\ 1\end{bmatrix} \tag{4-13}$$

把式（4-13）展开，可以得到

$$x_t=k_1+k_2l_2^t+k_3l_3^t \tag{4-14}$$

其中，k_1、k_2、k_3是待定的常数，它的值由该粒子初始位置所确定，由式（4-14），有

$$\begin{bmatrix}x_0\\ x_1\\ x_2\end{bmatrix}=\begin{bmatrix}1 & 1 & 1\\ 1 & l_1 & l_2\\ 1 & l_1^2 & l_2^2\end{bmatrix}\begin{bmatrix}k_1\\ k_2\\ k_3\end{bmatrix} \tag{4-15}$$

另外，由式（4-9），x_2可由x_1、x_0表示成

$$x_2=(1+w-f_1-f_2)x_1-wx_0+f_1p_i+f_2p_g \tag{4-16}$$

结合式（4-15）和式（4-16），可分别得到k_1、k_2和k_3的值

$$k_1=\frac{f_1p_i+f_2p_g}{f_1+f_2}$$

$$k_2=\frac{l_3(x_0-x_1)-x_1+x_2}{g(l_2-1)}$$

$$k_3=\frac{l_2(x_1-x_0)+x_1-x_2}{g(l_3-1)}$$

不难找到合适的参数 w 、 f_1 和 f_2 使 $\max(\|l_3\|,\|l_2\|)<1$，其中， l_3 、 l_2 可以是复数。

显然，当 $\max(\|l_3\|,\|l_2\|)<1$ 时

$$\lim_{t\to\infty} x_t = k_1 = a\, p_i + (1-a)\, p_g$$

其中， $a = \dfrac{f_1}{f_1+f_2}$ 。

证毕。

当 $x(t)$ 收敛时， $\lim\limits_{x\to+\infty} x(t) = k_1 = \dfrac{f_1 p_i + f_2 p_g}{f_1+f_2}$ ，以上都假设 f_1 、 f_2 是常数，但实际上 f_1 和 f_2 是 $[0,c_1]$ 和 $[0,c_2]$ 上均匀分布的随机数，其期望值 $E(f_1) = c_1\int_0^1 \dfrac{x}{1-0}\mathrm{d}x = \dfrac{c_1}{2}$ ， $E(f_2) = c_2\int_0^1 \dfrac{x}{1-0}\mathrm{d}x = \dfrac{c_2}{2}$ ，那么，尽管存在一些扰动，当 $\max(\|l_2\|,\|l_3\|)<1$ 时， $x(t)$ 收敛，且 $\lim\limits_{x\to+\infty} x(t) == \dfrac{\frac{c_1}{2}p_i + \frac{c_2}{2}p_g}{\frac{c_1}{2}+\frac{c_2}{2}} = \dfrac{c_1 p_i + c_2 p_g}{c_1+c_2}$ 。这表明粒子将收敛到其个体最优位置和全局最优位置连接线段上的某一点，而粒子的个体最优位置也趋向于全局最优位置，因此，粒子将最终收敛到全局最优位置。

图 4-6 中黑色区域表示满足不等式 $(1+w-f_1-f_2)^2 < 4w$ 的区域，即 g 为复数的区域；图 4-7 中黑色三角区域表示 $\max(\|l_2\|,\|l_3\|)>1$ 的区域，该区域满足不等式 $w < \dfrac{f_1+f_2}{2}-1$ ；灰色区域表示 $\max(\|l_2\|,\|l_3\|)<1$ 的区域，并且灰度级越高表示收敛越快；白色区域表示 $\max(\|l_2\|,\|l_3\|)=1$ 的区域，满足 $w = \dfrac{f_1+f_2}{2}-1$ 或 $f_1+f_2=0$ 。因此，只要取 $w > \dfrac{c_1+c_2}{2}-1$ ，就可以保证粒子群算法收敛。文献[87]给出了一组保证粒子群算法收敛的典型参数取值： $w=0.7298, c_1=c_2=1.49618$ 。

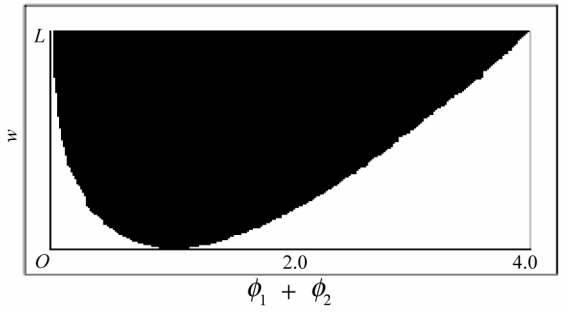


图 4-6　$g=\sqrt{(1+w-f_1-f_2)^2-4w}$ 为复数时的区域分布

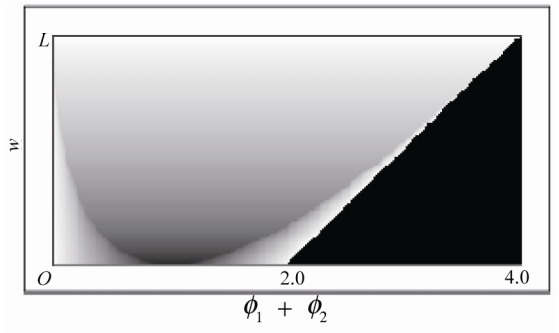


图 4-7　$\max\left(\|l_2\|,\|l_3\|\right)$ 取值分布

4.4　粒子轨迹的确定性分析

目前，关于 PSO 方法中粒子收敛性多采用确定性分析，此时，r_1、r_2 的数学期望为 1/2，令 $j_1=c_1/2$、$j_2=c_2/2$，$k=j_1+j_2=(c_1+c_2)/2$。在这种假设条件下，文献[117]做了相当有意义的工作。以下定理给出了粒子收敛的参数范围，是对从前研究成果的完善和补充。

定理 4-2　在确定性分析条件下，使粒子运行轨迹收敛的参数设置范围是直线 $k=2w+2$、$k=0$、$w=1$ 围成的三角形区域。

证明：在确定性条件下，式（4-1）可以改写成如下形式：

$$x_{t+2}-a\,x_{t+1}+wx_t-b=0 \qquad (4\text{-}17)$$

其中，$a=1+w-k$、$b=j_1p_i+j_2p_g$、$k=j_1+j_2$，显然式（4-17）为二阶线性差分方程，其特征方程为

$$l^2-a\,l+w=0 \tag{4-18}$$

通常根据式（4-18）解的不同情况进行讨论，令式（4-18）的判别式$D=a^2-4w=(1+w-k)^2-4w$，可分三种情况进行分析。

1．当$D>0$时

当$D>0$时，特征方程式（4-18）有两个实数解，此时差分方程式（4-17）有如下形式解：

$$x_t=C_1l_1^t+C_2l_2^t \tag{4-19}$$

其中，C_1、C_2是任意常数，l_1、l_2为特征方程的两个实根。由式（4-19）可以得出：当l_1、l_2的绝对值同时小于1时，x_t收敛。显然，当$w<0$或$w=0$且$k\neq1$时，$D>0$；当$w>0$时，$D=(1+w-k)^2-4w=(1+w-k+2\sqrt{w})(1+w-k-2\sqrt{w})$，故当$k>1+w+2\sqrt{w}$或$k<1+w-2\sqrt{w}$时，$D>0$，有

$$l_{1,2}=\frac{a\pm\sqrt{D}}{2}=\frac{a\pm\sqrt{a^2-4w}}{2} \tag{4-20}$$

不妨设$l_1=\dfrac{a-\sqrt{a^2-4w}}{2}$，$l_2=\dfrac{a+\sqrt{a^2-4w}}{2}$，当$l_1>-1$并且$l_2<1$时，$l_1$、$l_2$的绝对值同时小于1。不难求得若满足$l_1>-1$，必有$k<w+3$且$k<2w+2$；而满足$l_2<1$必有$k>0$且$k>w-1$；从而得到在$D>0$时的收敛条件。

2．当$D=0$时

当$D=0$时，特征方程有唯一实数解，此时差分方程式（4-17）有如下形式解：

$$x_t=C_1l^t+C_2tl^t \tag{4-21}$$

其中，C_1、C_2是任意常数，l为特征方程的根。当$|l|<1$时，x_t收敛。考查式（4-18）的判别式D，不难得出，当$w=0$且$k=1$时，$D=0$；而当$w>0$时，$k=1+w+2\sqrt{w}$或$k=1+w-2\sqrt{w}$时，$D=0$，此时有

$$l=\frac{a}{2}=\frac{1+w-k}{2} \tag{4-22}$$

不难求得若满足$|l|<1$，必有$k>w-1$且$k<w+3$；得到在$D=0$时的收敛条件。

3．当$D<0$时

当$D<0$时，特征方程式（4-18）有一对共轭复根，此时差分方程式（4-17）有如下形式解：

$$x_t=C_1R^t\cos tq+C_2R^t\sin tq \tag{4-23}$$

其中，C_1、C_2是任意常数，$R=\sqrt{w}$，当$R<1$时，式（4-23）收敛。考查式（4-18）的判别式D，可得当$1>w>0$，$k<1+w+2\sqrt{w}$或$k>1+w-2\sqrt{w}$时，$D<0$；式（4-23）收敛，得到情况$D<0$时的收敛条件。

综合上述 3 种情况，定理得证。

定理中的收敛条件如图 4-8 所示。

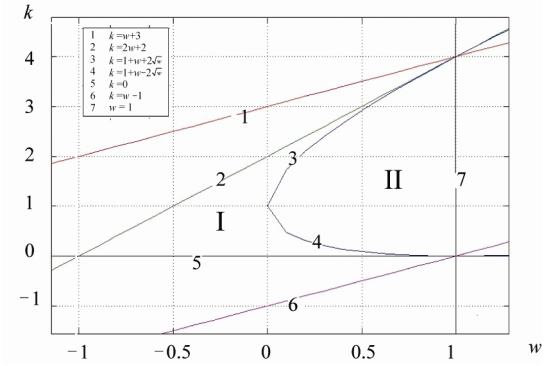


图 4-8　直线 2、5、7 围成的区间为收敛的参数范围

注：直线 1：k=w+3；直线 2：k=2w+2；曲线 3：k=1+2w+2$\sqrt{w}$；
曲线 4：1+2w-2$\sqrt{w}$；直线 5：k=0；直线 6：k=w-1；直线 7：w=1。

事实上，文献[117]中结论恰恰为这个三角形区域中$w>0$的那部分，由于拘泥于 PSO 模型中对w的解释，大部分研究忽略了$w<0$时的情况，抛开这些物理解释，将问题抽象出来，研究其本质，发现在$w<0$时依然存在合适的参数使粒子运行轨迹收敛。图 4-8 中使粒子轨迹收敛的三角形区域由两部分构成，分别对应$D>0$（区域 I）和$D\leqslant 0$（区域 II）两种不同情况，虽然参数设置在这个区间内都能使粒子收敛，但是收敛的方

式是不同的，这与不同情况下解的形式不同有关。

定理 4-2 所描述的参数设置范围只是必要条件，以此为基础，下面讨论随机条件下合理的参数设置范围。

4.5 粒子的分布特征

在实际执行过程中，粒子的运行环境和定理中假设存在如下差异：

① $j_1 = c_1 * \text{rand}(0,1)$ 、$j_2 = c_2 * \text{rand}(0,1)$ 不是常数而是随机数；

② 在运行过程中，特别是算法执行的早期全局最优 p_g 和个体最优 p_i 经常变化。

一些研究已经证明，在确定性分析前提下，粒子最终将收敛到点 u，其中 $u=(c_1p_i+c_2p_g)/(c_1+c_2)$。下面考查当参数设置满足定理 4-2 条件，且 $j_1 = c_1 * \text{rand}(0,1)$ 、$j_2 = c_2 * \text{rand}(0,1)$ 为随机数时，粒子的分布规律。

在许多研究中，通常取 $w = 0.729, c_1 = c_2 = 1.49445$，这组参数满足定理 4-2 中的条件。同时在大部分试验中，这组参数使 PSO 取得了较佳的搜索效果，考查在此参数设置下，粒子的分布情况。令 $x_{i0}, x_{i1}, x_{i2}, \mathbf{L}, x_{iT}$ 表示粒子 x_i 在时间 T 内经历的有位置，将[-10| p_g - p_i |，10| p_g - p_i |]划分成 200 个相等的小区间，每个小区间的距离是| p_g - p_i |/10。图 4-9 中有两条曲线，其中一条反映的是在上述参数设置下，每个小区间内粒子平均个数曲线；另一条曲线则是以 $u=(c_1p_i+c_2p_g)/(c_1+c_2)$为中心，$\sigma$=1.4124 (| p_g - p_i |)/2 为方差的正态分布曲线。

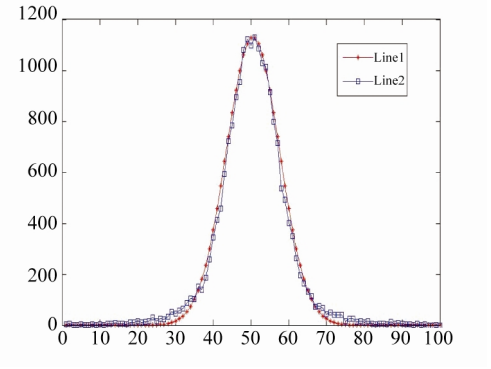


图 4-9　粒子分布曲线与正态分布曲线

可以看出这两条曲线是基本吻合的，对于其他满足定理 4-2 条件的参数设置，粒子也表现出以上分布规律，不同的是：不同的参数设置对应于正态分布的方差是不同的。特别是当参数设置接近定理 4-2 所述区间边界时，方差变化速度较大且不稳定。

在同一组参数的前提下，p_g 和 p_i 之间的距离越小，方差越小。p_i 和 p_g 的每一次变化都意味着有新的全局最优的发现或新的搜索区域的开辟，此时粒子将重新调换位置，按原分布规律进行搜索。

对于那些方差过大或发散的参数设置，并不是我们所关心的。这里需要把握的是什么样的参数设置能使粒子保持合理的分布。下面通过粒子的聚度来进行讨论。

4.6　粒子的聚度

定义 4-1　设 $x_{i0}, x_{i1}, x_{i2}, \mathbf{L}, x_{iT}$ 表示粒子 x_i 在 w、c_1、c_2 及 p_g 和 p_i 条件下，时间 T 内经历的所有位置，定义 $x_{i0}, x_{i1}, x_{i2}, \mathbf{L}, x_{iT}$ 关于 w、c_1、c_2 的聚度

$$k = N / TP \tag{4-24}$$

其中，$N = \mathrm{num}(x_j \in (L, H))$（$j = 0,1\mathbf{L},T$；$L = \min(p_i, p_g)$；$H = \max(p_i, p_g)$）为落在 p_i 和 p_g 之间粒子的个数，$P = \dfrac{1}{s\sqrt{2\pi}} \int_{-s}^{s} \mathrm{e}^{\frac{-(t-u)^2}{2s^2}} \mathrm{d}t$，$u = (p_i + p_g)/2$，$s = \dfrac{|p_g - p_i|}{2}$。

从定义中可以看出：聚度 k 的分母部分表示的是 $x_{i0}, x_{i1}, x_{i2}, \mathbf{L}, x_{iT}$ 在以 u 为中心、s 为方差时，期望落在 p_i、p_g 之间粒子的个数，聚度 k 反映的是在 w、c_1、c_2 下实际落在 p_i、p_g 之间粒子的个数和这个期望值的比。

当 T 值充分大时，k 值仅与参数 w、c_1、c_2 有关，表 4-1 说明了一组参数设置下 p_i、p_g 的改变与 k 值的变化情况。在同一组参数设置下当 p_i、p_g 发生变化时聚度值稳定。

表 4-1　相同参数下，不同 p_i、p_g 的粒子聚度

p_i 、p_g	w=0.7, c_1=1.4, c_2=1.4					
	1	2	3	4	5	平均植
p_i = -10, p_g =10	0.95896	0.96461	0.96970	0.95014	0.96890	0.96246
p_i = -0.1, p_g =0.1	0.96334	0.96490	0.96294	0.95899	0.95483	0.96100
p_i = -0.001, p_g =0.001	0.96775	0.95417	0.96214	0.95888	0.95950	0.96049

由于聚度与 p_i、p_g 无关，因此，可以通过聚度与算法搜索性能的关系来确定 w 、c_1 、c_2 的合理范围。在参数设置接近定理 4-2 所描述区间的边界时，聚度表现不稳定，这表明在这种参数条件下，粒子运行轨迹对随机扰动非常敏感。但这种情况和发散的情况都不是我们所关心的。

通常情况下，什么样的聚度能够获得最好的结果，对于不同的问题会有所不同，但有一点是可以肯定的：如果粒子的聚度过小，会使粒子漫布到整个搜索空间，不能形成有效的搜索，如果粒子的聚度过大，就会使粒子击中分布在群体最优和个体最优之间，形成早熟。图 4-10 给出了当聚度 κ=0.5 和 κ=1 两条曲线和相关数据。一般情况下，参数设置应使聚度保持在这一范围内。

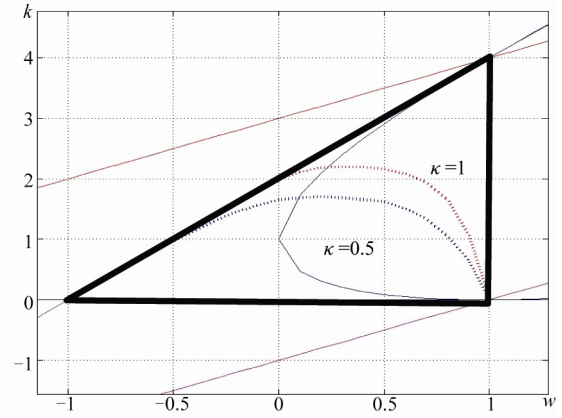


图 4-10　合理的聚度范围

如表 4-2 所示为图 4-5 中聚度曲线的相关数据。

表 4-2　图 4-5 中聚度曲线的相关数据

w	$k=(c_1+c_2)/2$	
	κ=1	κ=0.5
-0.9	0.1975	0.2
-0.8	0.3950	0.4
-0.7	0.5851	0.6
-0.6	0.7756	0.8
-0.5	0.9660	1
-0.4	1.1400	1.2
-0.3	1.3101	1.4
-0.2	1.4559	1.6
-0.1	1.5611	1.8
0	1.6512	2
0.1	1.6900	2.1100
0.2	1.7051	2.1851
0.3	1.6909	2.1855
0.4	1.6584	2.1951
0.5	1.6212	2.1621
0.6	1.5123	2.0849
0.7	1.3351	1.9383
0.8	1.0522	1.6563
0.9	0.6589	1.0828
0.99	0.0681	0.1431

从图 4-5 中可以看出，当 $w<-0.5$ 时，$k=1$ 和 $k=0.5$ 两条曲线和 $k=2w+2$ 基本重合，这充分说明在这一区间内寻找合适的参数是相对困难的。经常采用的参数 $w=0.729$，$c_1=c_2=1.49445$，聚度值为 $k=0.8280$。

聚度从一定意义上反映了粒子在搜索过程中的分布情况，只有合理的参数设置使粒子的聚度值不是特别大和特别小的时候，PSO 方法才有可能搜索到最好的结果。

这里还要注意，聚度的定义是以粒子的群体最优和全局最优固定为条件的，聚度不因为群体最优和全局最优动态改变而变化，不合理的聚度在 p_g、p_i 固定不变的前提下造成粒子分布的分散或过度集中，因此，当 p_g、p_i 动态变化时，更不可能得到理想的搜索

效果。所以，设置合理的参数，使粒子的聚度值合理，同样是 PSO 方法取得较好的必要条件。

PSO 在实际运行条件下，由于 p_g、p_i 动态变化，此时不仅要考查聚度值是否合理，还要考虑粒子收敛的速度是否合理，在 PSO 方法中，粒子之间是通过修正全局最优 p_g 来相互影响的，若粒子运行速度收敛得过快，直接造成粒子的修正量 v 迅速减小，从而使得群体的全局最优 p_g 和每个粒子个体最优 p_i 之间的距离较小，根据上面的分析可知，粒子是以 $m=\frac{c_1p_i+c_2p_g}{c_1+c_2}$ 为中心、$s=a(\frac{|p_g-p_i|}{2})$ 为方差呈正态分布，p_g 和 p_i 之间距离较小，又会进一步造成修正量 v 的减小，即使 p_g 和 p_i 之间能够有合理聚度，粒子速度 v 的过快衰减也会造成 p_g 和 p_i 的迅速接近而形成早熟现象。

4.7 模拟退火粒子群优化方法

在标准 PSO 方法中没有选择操作，粒子一如既往地按照本身的“飞行”方式进行移动，即使粒子的下一个位置的评价函数值很差，粒子依然用这个位置代替当前位置，虽然 PSO 可以通过参数 c_1、c_2、w、c 控制粒子向最优位置靠拢，但参数的选择是细微的，特别是在算法执行的最后阶段，这时粒子评价函数值相对接近，参数的微小差异和任何一点儿随机扰动都会给运行结果带来一定的影响，使粒子容易跳出最优解附近的某一邻域，这就在一定程度上使 PSO 方法在局部“搜索”能力上表现较差。

针对上述情况，在 PSO 中引入了模拟退火思想，该思想最早是由 N.Metropolis 等人提出的，该思想用于解决优化问题的出发点是基于物理中固体物质的退火过程与一般组合优化问题间的相似性。在对固体物质进行退火处理时，通常先将它加温融化，使其中的分子可以自由运动，然后，随着温度的逐渐下降，分子也逐渐形成低能态的晶格。若温度下降速率足够慢，则固体一定会形成最低能量基态。对于优化问题来说，它也有类似的过程。

4.7.1　模拟退火

模拟退火算法来源于固体退火原理，将固体温度加热至充分高，再让其慢慢冷却，加温时，固体内部粒子随温度升高变为无序状态，内能增大；而慢慢冷却时，粒子渐趋有序，在每个温度都达到平衡态；最后，在常温时达到基态，内能减为最小。根据 Metropolis 准则，粒子在温度 T 时趋于平衡的概率为 $e^{-\Delta E/(kT)}$，其中，E 为温度 T 时的内能，ΔE 为其改变量，k 为 Boltzmann 常数。用固体退火模拟组合优化问题，将内能 E 模拟为目标函数值 f，温度 T 演化成控制参数 t，即得到解组合优化问题的模拟退火算法，模拟退火的基本过程如算法 4-2 所示。

算法 4-2　模拟退火方法

0	Algorithm：Simulated Annealing SA
1	初始化：初始温度 T（T 要充分大），初始状态 S ，每个 T 值的迭代次数 L；
2	对 $k=1,\cdots,L$ 执行 Step3 至 Step6；
3	产生新解 S'；
4	计算增量 $\Delta t'=C(S')-C(S)$，其中 $C(S)$为评价函数；
5	若 $\Delta t'<0$，则接受 S'作为新的当前解，否则，以概率 $\exp(-\Delta t'/T)$接受 S'作为新的当前解；
6	如果满足终止条件则输出当前解作为最优解，结束程序；否则，转到 Step7；
7	令 T 逐渐减少，转到 Step2。

模拟退火算法求得的解与初始状态 S（是算法迭代的起点）无关；模拟退火算法具有渐近收敛性，已在理论上被证明是一种以概率 1 收敛于全局最优解的全局优化算法。

模拟退火算法的应用很广泛，可以较高的效率求解最大截问题（Max Cut Problem）、0-1 背包问题（Zero One Knapsack Problem）、图着色问题（Graph Colouring Problem）、调度问题（Scheduling Problem）等。

关于模拟退火算法的参数控制，其主要问题有以下 3 点。

1．温度 T 的初始值设置问题

温度 T 的初始值设置是影响模拟退火算法全局搜索性能的重要因素之一，初始温度越高，则搜索到全局最优解的可能性越大，但因此要花费大量的计算时间；反之，则可

节约计算时间，但全局搜索性能可能受到影响。实际应用过程中，初始温度一般需要依据实验结果进行若干次调整。

2．退火速度问题

模拟退火算法的全局搜索性能也与退火速度密切相关。一般来说，同一温度下的“充分”搜索（退火）是相当有必要的，但这需要计算时间。实际应用中，要针对具体问题的性质和特征设置合理的退火平衡条件。

3．温度管理问题

温度管理问题也是模拟退火算法难以处理的问题之一。实际应用中，由于必须考虑计算复杂度的切实可行性等问题，常采用如下所示的降温方式：

$$T(t+1)=kT(t)$$

其中，k 为正的略小于 1.00 的常数，t 为降温的次数。

4.7.2 模拟退火粒子群优化

把上述思想引入到 PSO 方法中，形成 PSOwSA 方法，其执行过程如算法 4-3 所示。

算法 4-3 模拟退火粒子群优化算法

0	Algorithm：Particle Swarm Optimization with Simulate Annealing, PSOwSA
1	Initialize population and parameters T, T_L, a /*其中，$T_L < T, a < 1$ */
2	Do
2.1	For i = 1 to Population Size
2.1.1	If $f(x_i) < f(p_i)$ then $p_i = x_i$， $p_g = \min(p_i)$
2.1.2	For d = 1 to Dimension
2.1.2.1	$v_{\text{temp}\_d} = v_{id} + c_1 j_1 (p_{id} - x_{id}) + c_2 j_2 (p_{gd} - x_{id})$
2.1.2.2	$x_{\text{temp}\_d} = x_{id} + v_{\text{temp}\_d}$
2.1.2.3	if $x_{\text{temp}\_d} < a_d$ then $x_{\text{temp}\_d} = a_d$; if $x_{\text{temp}\_d} > b_d$ then $x_{\text{temp}\_d} = b_d$
2.1.3	End for
2.1.4	if $f(x_{\text{temp}}) \leqslant f(x_i)$ then $x_i = x_{\text{temp}}$， else $x_i = x_{\text{temp}}$ with Probability: P_T

续表

算法 4-3 模拟退火粒子群优化算法	
2.2	End for
2.3	if termination criterion is met then stop
2.4	$T = aT$
3	Until $T \leqslant T_L$

算法 4-3 中，设置粒子参数，使 $\max(\|l_3\|,\|l_2\|)<1$，估计每一个函数的最大值 $f_{i\_\max}$ 和最小值 $f_{i\_\min}$，令初始温度 $T=10(f_{i\_\max}-f_{i\_\min})$，且 $P_T=\exp(-(f(x_{\text{temp}})-f(x_i))/T)$。研究 PSOwSA 方法，会发现粒子在运动过程中，当粒子下一个位置比当前位置好时，粒子移动到下一个位置；反之，若下一个位置比当前位置差，则粒子并不直接移动到下一个位置，而是以某一概率进行移动，且通过温度 T 来控制这一概率。从直观上，若把标准 PSO 看成对鸟类觅食等群体行为的简单模拟，则在 PSOwSA 中的“鸟”更加聪明，它不是盲目地直接扑向下一个位置，而是以某种概率“试探”后再行动。因此，当温度下降的足够慢时，粒子不会轻易跳出有“希望”的搜索区域，从而增强了粒子的局部搜索能力。

认真研究上述 PSOwSA 算法的执行过程，并结合定理 4-1，可得出以下推论。

推论 4-1　在定理 4-1 假设条件下，当 $t\to\infty$ 时，PSOwSA 算法中每个粒子将最终收敛到以个体最优和群体最优为端点的线段上。

结合定理 4-1 和算法的具体描述，容易证明以上推论。

在标准 PSO 的运行过程中，粒子 x_i 直接用 $t+1$ 步位置 $x_i(t+1)$ 取代 $x_i(t)$，而不考虑这两点评价函数值的好坏。在算法的运行过程中，特别是在算法运行的后期，粒子的个体最优和群体最优之间的距离变得很近时，粒子容易跳出原来的搜索区域，造成粒子局部搜索能力下降。PSOwSA 引入“模拟退火”的思想来克服这一不足，在 PSOwSA 的运行过程中，粒子 x_i 在第 $t+1$ 步时按照某一概率用 $x_i(t+1)$ 取代 $x_i(t)$，同时采用温度 T 来控制这一概率，温度 T 随着算法的执行缓慢下降，此时，若 $x_i(t+1)$ 的评价函数值差于 $x_i(t)$ 的评价函数值，用 $x_i(t+1)$ 取代 $x_i(t)$ 的概率不断减小，从而控制粒子使之不能从有“希望”的搜索区域中跳出。在这里要说明的是：只有当温度 T 下降的速度充分缓慢时，才

能达到理想搜索效果，若温度下降的太快，会使粒子移动的概率迅速变小，造成粒子在某一搜索区域内停滞不前。下面的实验数据也说明了这一点。

4.8 有分工策略的粒子群优化方法

设搜索空间 $S=[a_1,b_1]\times\mathbf{L}\times[a_D,b_D]$，POP 是由 N 个 S 上的粒子组成的粒子群，$f:S\to R$，是评价函数，x 是 POP 中的一个粒子，并且对于 $\forall y\in$ POP 都有 $f(x)\leqslant f(y)$（对极小值优化问题），把 POP 中所有满足上述条件的粒子所构成的集合记为 C(POP)。显然，C(POP) 中至少存在一个元素，有些时候 C(POP) 可能会有多个元素，这与函数 f 的性质有关。

设粒子群 POP 由 N 个粒子组成，且最初由算法随机产生。把粒子群 POP 划分为 3 个互不交叉的子群体并分工。

1．子群 POP_Core

若 C(POP) 中的元素个数超过 $N/3$，则在 C(POP) 中随机选取 $N/3$ 个粒子构成 POP_Core，否则，POP_Core=C (POP)。在迭代过程中对集合 POP_Core 采用 EP 的思想，即集合中的所有个体 x_i 都按公式 $x'_{id}=x_{id}+s\,N(0,1)$（$d=1,\mathbf{L},D$）进行变异，其中，s 为变异步长，N [0,1]为[0,1]上满足正态分布的随机数，变异过程中变异步长 s 按 1/5 成功法则进行自适应调整，即 $s=ts$，当变异成功概率大于 1/5 时，$t>1$；当变异成功概率等于 1/5 时，$t=1$；当变异成功的概率小于 1/5 时，$t<1$。假设 POP_Core 中有 m 个元素，变异后又产生了 m 个元素，在这 $2m$ 个元素中选择出最好的 m 个元素重新构成下一代 POP_Core。

2．子群 P_Near

设 POP_Core=$\{\hat{x}_1,\hat{x}_2,\mathbf{L},\hat{x}_m\}$，令 Core_Centre=$\sum\hat{x}_i/m$，在 POP-POP_Core 中选择 ($N$-$m$)/2 个距离 Core_Centre 相对较近的粒子构成的集合 P_Near，在迭代的过程中 P_Near 中粒子按算法 4.1 执行，选择适当参数 w、c_1、c_2，使 $\max(\|l_3\|,\|l_2\|)<1$。

3．子群 P_Far

令 P_Far=POP-POP_Core-P_Near，显然 P_Far 是由距离 Core_Centre 相对较远的粒子构成的集合，在迭代过程中 P_Far 中粒子同样按算法 4-1 执行，只是当粒子修正后的位置在某一维度上超过搜索范围时，进行了反弹处理，而不是直接取它在这一维度上的上界或下界。为 P_Far 中的粒子选择适当参数 w、c_1、c_2，使 $\max(\|l_3\|,\|l_2\|)>1$。算法 4.4 具体描述了 PSOwDOW 方法。

算法 4.4 中，$\left\lfloor \frac{a_d - x_{id}}{b_d - a_d} \right\rfloor$ 表示 $\frac{a_d - x_{id}}{b_d - a_d}$ 的整数部分。PSOwDOW 方法通过这种“分工”的策略，把整个群体分成 3 个互不交叉的子群体，并且每个子群体承担着不同的搜索任务，这种改进不但继承了标准 PSO 方法“开拓能力”强这一优点，同时增强了算法的局部“搜索”能力。

推论 4-2　在定理 4-1 假设条件下，PSOwDOW 算法中，子群体 POP_Core 始终是群体最优粒子的集合，P_Near 中的粒子当 $t\to\infty$ 时将收敛到以 POP_Core 的中心 Core_Centre 和个体最优 p_i 为端点的线段上，子群体 P_Far 中的粒子运行轨迹发散。

值得注意的是，定理 4-1 和两个推论都是建立在一定假设基础之上的，而这个假设与实际情况存在着一定的差别。首先，p_i 和 p_g 在算法执行过程中不可能保持不变；其次，$f_1=c_1r_1$、$f_2=c_2r_2$ 不是常数，而是随机数。事实上，当 $f_1=c_1r_1$、$f_2=c_2r_2$ 为随机数时，若粒子的参数使 $\max(\|l_3\|,\|l_2\|)<1$，在 p_i（或 Core_Centre）和 p_g 在一段时间内保持不变的情况下，粒子在这段时间内，将在以这两点为对角线的超立方体内呈正态分布。这表明当粒子的参数使 $\max(\|l_3\|,\|l_2\|)<1$ 时，粒子将根据自己本身的搜索经验和整个群体的搜索经验在最有“希望”的区域内进行搜索。在上述过程中，粒子会不断地发现新的个体最优 p_i 或群体最优 p_g，当 p_i（或 Core_Centre）和 p_g 发生变化时，粒子将重新调整自己的运行轨迹及收敛趋势，继续按上述规律在空间内搜索，随着 p_i 和 p_g 间距离的不断缩小，以这两点为对角线的超立方体也不断缩小。

	算法 4-4　有分工策略的粒子群优化方法
0	Algorithm:Particle Swarm Optimization with division of work, PSOwDOW
1	Initialize population: POP and all parameters, Core_Mutation= ϕ
2	Do
2.1	Divide POP into subswarm: POP_Core, *P*_Near and *P*_Far
2.2	Compute Core _ Centre = $\sum_{k=1}^{m} \hat{x}_k / m$， 其中 POP _ Core = $\{\hat{x}_1, \mathbf{L}, \hat{x}_m\}$
2.3	For i = 1 to Population Size
2.3.1	If x_i in POP_Core
2.3.1.1	For d = 1 to Dimension
2.3.1.1.1	$x'_i = x_{id} + s\ N(0,1)$
2.3.1.2	End for
2.3.1.3	Core _ Mutation = Core _ Mutation $\mathbf{U}$ x'_i
2.3.2	If x_i in POP_Near
2.3.2.1	Set parameters，let $\max(\|l_3\|, \|l_2\|) < 1$
2.3.2.2	For d = 1 to Dimension
2.3.2.2.1	$v_{id} = v_{id} + c_1 j_1 (p_{id} - x_{id}) + c_2 j_2 (\text{Core\_Centre}_d - x_{id})$
2.3.2.2.2	$x_{id} = x_{id} + v_{id}$
2.3.2.2.3	if $x_{id} < a_d$ then $x_{id} = a_d$ if $x_{id} > b_d$ then $x_{id} = b_d$
2.3.2.3	End for
2.3.3	If x_i in POP_Far
2.3.3.1	Set parameters，let $\max(\|l_3\|, \|l_2\|) > 1$
2.3.3.2	For d = 1 to Dimension
2.3.3.2.1	$v_{id} = v_{id} + c_1 j_1 (p_{id} - x_{id}) + c_2 j_2 (Core\_Centre_d - x_{id})$
2.3.3.2.2	$x_{id} = x_{id} + v_{id}$
2.3.3.2.3	if $x_{id} < a_d$ then $x_{id} = 2a_d - x_{id} - \lfloor (a_d - x_{id}) / (b_d - a_d) \rfloor (b_d - a_d)$
2.3.3.2.4	if $x_{id} > b_d$ then $x_{id} = 2b_d - x_{id} + \lfloor (x_{id} - b_d) \setminus (b_d - a_d) \rfloor (b_d - a_d)$
2.3.3.3	End for
2.4	End for
2.5	Adjust σ in term of 1/5 success rule
2.6	Reconstruct POP_Core From POP_Core $\mathbf{U}$ Core_Mutation
2.7	POP=POP_Core $\mathbf{U}$ *P*_Near $\mathbf{U}$ *P*_Far
3	Until termination criterion is met

在 PSOwDOW 算法中采用“分工”的策略来克服标准 PSO 方法存在的不足，整个群体被分成 3 个子群体：POP_Core、*P*_Near 和 *P*_Far。其中，POP_Core 采用 EP 的搜索策略，不断地在群体最优附近探索新的群体最优，从而保证了群体最优解附近的充分搜索；*P*_Near 中粒子参数使 $\max(\|l_3\|,\|l_2\|)<1$，因此，粒子的搜索范围是 Core_Centre 与粒子个体最优 p_i 之间更广泛的区域，通常情况下，这一区域被认为是最有希望发现群体最优和个体最优的区域；*P*_Far 中粒子参数使 $\max(\|l_3\|,\|l_2\|)>1$，因此，粒子运行轨迹发散，它们在算法执行过程中承担着开辟新搜索区域的任务，这部分粒子使群体充分保持了个体的多样性，避免算法“早熟”现象的发生。在 PSOwDOW 执行过程中，每个粒子的角色不是一成不变的，这使 3 个子群体中的元素也不断发生变化，*P*_Near、*P*_Far 中的粒子可能在下一步成为 POP_Core 中的成员，这时往往意味着又有新的搜索区域被开辟。在算法的执行过程中，不断有新群体最优和个体最优被发现，直到满足终止条件。

PSOwSA 和 PSOwDOW 通过上述两种改进策略来克服标准 PSO 方法中的不足，这也是 PSOwSA 和 PSOwDOW 两种方法的最终搜索结果较标准 PSO 方法得以提高的原因。

4.9　算法测试

对以下 4 个测试问题采用 PSOwSA 方法，运行 50 次所得测试结果如表 4-3 所示。

表 4-3　PSOwSA 运行 50 次所得最优结果平均值

函数	维数	α	单个粒子移动次数	PSOwSA 平均最优结果
f_1	30	0.65	2000	7.5832
		0.80	5000	0.2571
		0.90	10000	<10e-5
f_5	30	0.65	2000	2068.58
		0.80	5000	947.338
		0.90	10000	378.265
f_9	30	0.65	2000	49.7761
		0.80	5000	44.9864
		0.90	10000	42.1748

续表

函数	维数	α	单个粒子移动次数	PSOwSA 平均最优结果
f_{11}	30	0.65	2000	0.4332
		0.85	5000	0.1104
		0.90	10000	0.0097

PSOwSA 方法比标准 PSO 方法多一层循环结构，因此，用“单个粒子移动次数”来粗糙地衡量算法的计算量。在 PSOwSA 方法中，假设粒子在某个温度下的“飞行”次数为 K，可以得出每个粒子修正次数的估计值。因此，参数的值影响每个粒子的移动次数，当值较小时，温度下降很快，粒子的移动次数也就较小，这时，容易造成粒子在某一区域内停滞不前，影响算法的最后执行结果；反之，当值较大时，温度下降缓慢，欲达到最小温度，粒子的移动次数相应变大，这时，算法的计算量就会相应增大。在表 4-3 中，当 $a=0.65$ 时，温度下降较快，这时 PSOwSA 与标准 PSO 方法所搜索到的结果大致相同，而随着值的增大，当 $a=0.90$ 时，温度下降变得缓慢，此时 PSOwSA 的平均搜索结果得到较大改善，这是靠简单的提高标准 PSO 方法的迭代次数所不能达到的。这说明 PSOwSA 方法当温度下降充分缓慢时，在一定程度上克服了标准 PSO 方法在局部搜索能力上的不足。

同样针对表 4-3 中的 4 个问题，采用 PSOwDOW 方法，运行 50 次所得测试结果如表 4-4 所示。

表 4-4　PSOwDOW 方法运行 50 次所得最优结果平均值和不同子群体搜索到群体最优次数的平均值

函数	维数	单个粒子移动次数	POP_Core 搜索到群体最优的平均次数	P_Near 搜索到群体最优的平均次数	P_Far 搜索到群体最优的平均次数	PSOwDOW 平均最优结果
f_1	30	2000	171	56	19	<10e-8
f_5	30	2000	77	15	7	79.3358
f_9	30	2000	33	9	3	44.1858
f_{11}	30	2000	95	22	9	<10e-5

结合表 4-3 和表 4-4 可以看出，相同迭代次数的前提下，PSOwDOW 方法得到的最优结果的平均值较标准 PSO 方法所获结果有很大的提高，若提高 PSOwDOW 方法的运行次数，所得结果更令人满意，这也是通过增加标准 PSO 方法的迭代次数所不能达到的。

从表 4-4 中可以看出，子群体 POP_Core 搜索到群体最优解的次数明显多于另外两个子群体，但并不能因此忽略这两个子群体的作用，*P*_Near 扩展了 POP_Core 的搜索范围，*P*_Far 开辟了新的搜索区域。虽然 *P*_Near 和 *P*_Far 中的粒子搜索到群体最优解的次数占的比例很小，但这对算法有着非常重要的意义，它能使 POP_Core 摆脱局部最优的吸引，尽可能地避免了算法“早熟”现象的发生。通过测试表明，标准 PSO 方法中的不足，在 PSOwDOW 中得到了较好的弥补。

就所得的最优结果的平均值而言，PSOwDOW 更接近问题的全局最优解，而且在 50 次运行中，所得最好结果和最差结果相差不大，这说明 PSOwDOW 方法相对稳定，这对求解优化问题也十分重要。PSOwSA 方法运行 50 次所得最优解的平均值比 PSOwDOW 方法所得最优解平均值略差，但在充分运行的条件下，即当温度下降充分慢时 PSOwSA 同样能搜索到令人满意的结果，这时，计算开销是首先要考虑的问题。从直觉上 PSOwSA 的性能仿佛还有提高的潜力，参数选择是提高该算法的关键，同时也是一个难点，这是因为无论是模拟退火还是 PSO 方法的参数选择都很复杂，我们将在这方面加以尝试。

PSOwSA 和 PSOwDOW 这两种改进策略都不同程度地改善了标准 PSO 方法局部搜索能力较弱的事实。PSOwSA 方法是在 PSO 方法的基础上引入模拟退火思想，当温度变化相对缓慢时，能够搜索到较好的结果，而增加标准 PSO 方法的迭代次数不能收到这样的效果。PSOwDOW 方法是在 PSO 中加入分工的思想，把整个粒子群分成 3 个子群体，每个群体都有不同的分工，在执行过程中这 3 个群体不断地发生变化。这种处理方式较好地平衡了局部“搜索”和“开拓”新领域之间的关系，在计算量没有明显增加的前提下，计算结果却有很大的提高，同时算法相对稳定，即在 50 次运行中所得最好结果和最差结果相差不大。综上所述，这两种方法都具有各自的特点，在解决复杂非线性优化问题上有较强的应用价值。

4.10 动态优化

试验证明了 PSOwDOW 和 PSOwSA 能够成功地应用于静态优化问题，但由于受一些不确定因素的干扰，在现实世界中几乎很少系统是静态的。因此，优化算法是否有能力响应外部环境的动态改变，是衡量算法可用性的一项重要标准。文献[90]～[94]作了许多这方面的研究。为考查 PSOwDOW 和 PSOwSA 方法的跟踪动态最优解的能力，在 Sphere 函数的基础上，构造了 3 种类型的动态优化环境，在动态优化环境下对两种改进方法进行测试。

考虑三维 Sphere 函数： $f(x,y,z)=x^2+y^2+z^2$，这是一个只有唯一全局最优的凸函数，全局最优点位于（0, 0, 0），此时 $f(x,y,z)=0$。令三维 Sphere 函数 $f(x,y,z)$ 的全局最优点按不同的轨迹周期或随机地移动，构成下述 3 种不同类型的动态优化模型[90]：线性模型、环形模型、随机模型。

4.10.1 线性模型

在线性动态模型中，全局最优解在每一维上的偏移量，按式（4-25）计算：

$$d_k^t = d_k^{t-1} + s \quad d_k^0 = 0\ ,\ k = 1,2,3 \tag{4-25}$$

此时，函数最优解移动的轨迹为一条直线，如图 4-11 所示。

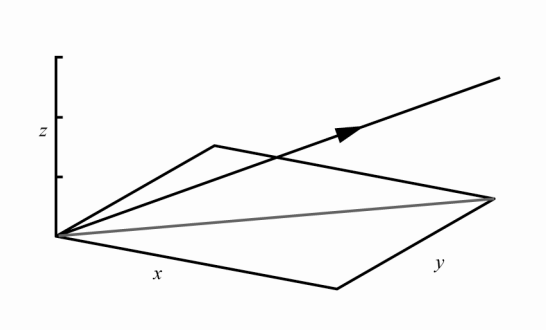


图 4-11 线性模型

4.10.2　环形模型

在线性动态模型中，全局最优解在每一维上的偏移量，按式（4-26）计算：

$$\begin{aligned} d_k^t &= d_k^{t-1} + s\sin\frac{2\pi t}{T} \quad k=1,3 \\ d_k^t &= d_k^{t-1} + s\cos\frac{2\pi t}{T} \quad k=2\ ,\ d_k^0 = 0 \end{aligned} \tag{4-26}$$

此时，函数最优解的轨迹为空间内的闭合曲线，如图 4-12 所示，且以 T 为周期。

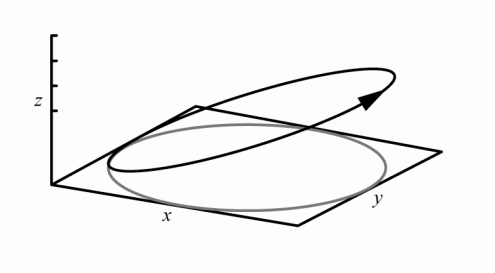


图 4-12　环形模型

4.10.3　随机模型

在线性动态模型中，全局最优解在每一维上的偏移量，按式（4-27）计算：

$$d_k^t = d_k^{t-1} + sN(0,1) \quad d_k^0 = 0\ ,\ \ k=1,2,3 \tag{4-27}$$

此时，函数最优解的轨迹如图 4-13 所示，在空间内随机移动。其中，s 是正常数。

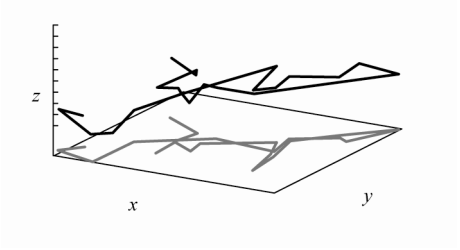


图 4-13　随机模型

由此，三维 Sphere 函数 $f(x,y,z)=x^2+y^2+z^2$ 转化为动态函数：$f(x,y,z,s,t)=(x-d_1^t)^2+(y-d_2^t)^2+(z-d_3^t)^2$。在这里若 t=m，则表示每间隔时间 m，$f(x,y,z,s,t)$ 的全局最优解就要发生一次移动。

4.10.4 动态优化仿真

研究算法对动态环境的适应性，首先要考虑算法如何探测到优化函数的改变。在本书中，采取单点探测策略，即随机选择空间内的某点（Sentry），并在算法执行中，每次迭代都对该点进行重新计算，当该点的评价函数值发生改变时，意味着评价函数发生了改变。这时，每个粒子都要进行重新评价。关于对动态环境的探测策略，本节不作更多的讨论。

令搜索空间为$[-50,50]^3$，对线性模型而言，当偏移量超过这个范围时，即当$d_k^t \geqslant 50$时，令$d_k^{t+1} = d_k^t - s$，$k = 1,2,3$，直到$d = 0$时，再令$d_k^{t+1} = d_k^t + s$，$k = 1,2,3$，如此循环；在随机模型中，当偏移量在某一维上大于 50 或小于 0 时，令$d = 50$或 0，而在环形模型中，控制步长使最优解的偏移量始终在这个范围内。取不同的步长 S 和动态控制控制参数 t，这里，最大的迭代次数为 1000。图 4-14～图 4-26 所示分别为在不同动态条件，运行 PSO 和 PSOwDOW 方法 50 次，算法搜索到的结果与最优解之间误差的平均值曲线。

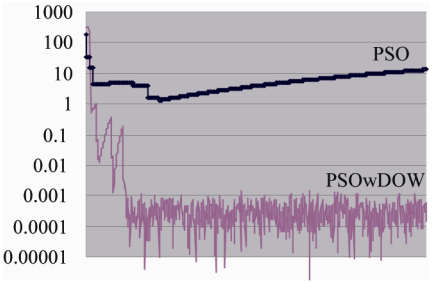


图 4-14　线性模型s=0.01，t=1

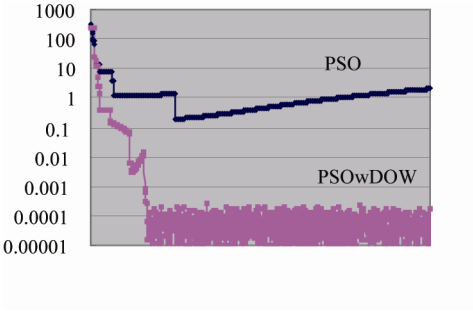


图 4-15 线性模型s=0.01，t=5

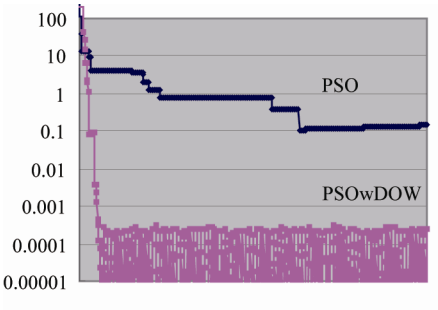


图 4-16　线性模型s=0.01，t=10

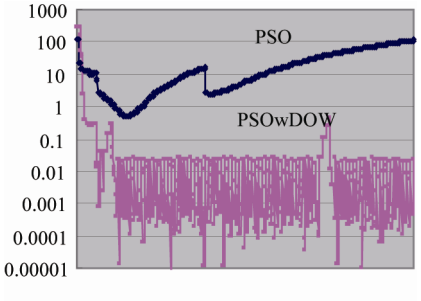


图 4-17　线性模型s=0.1，t=10

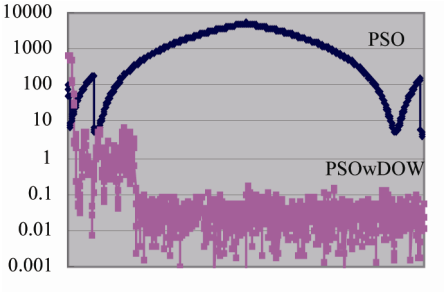


图 4-18　线性模型s=0.1，t=1

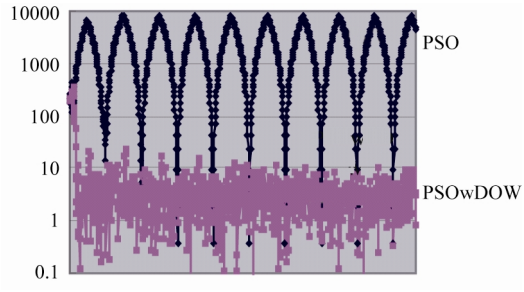


图 4-19　线性模型s=1，t=1

图 4-14～图 4-16 展示了 PSO 方法和 PSOwDOW 方法在线性动态模型中，当最优解修正步长 s=0.01，动态控制参数 t 分别为 1、5 和 10 时两个算法的误差曲线。当 t=1 和 t=5 时（见图 4-14、图 4-15），函数变化的频率较快，随着算法的不断执行，PSO 方法的误差值不断增大，而当 t=10 时，误差值趋于稳定，说明这时标准 PSO 方法有较充分的时间来响应动态函数的变化。同样，令 t=10，增大 s=0.1（见图 4-17），这时 PSO 方法的误差随迭代次数的增大而逐渐增大。这表明只有当动态函数的扰动周期 t 和修正步长 s 相对较小时，标准 PSO 方法才能有效地跟踪全局最优解的这种变化。在线性模型中考查 s=0.1、t=1 时的情形（见图 4-18），此时全局最优点从（0, 0, 0）移动到（50, 50, 50），然后又从（50, 50, 50）移动回（0, 0, 0），从图 4-15 中明显看出，PSO 方法已经无力跟踪这种步长和频率都相对较大的扰动，在算法执行的前 500 次误差值逐渐增大，而在算法执行的后 500 次，误差值却逐渐减小，但这并不是因为 PSO 方法追踪到了全局最优，而是因为在全局最优解从（50, 50, 50）向（0, 0, 0）移动的过程中，粒子群也恰恰在这附近搜索，当 s=1，t=1 时，全局最优在（0, 0, 0）和（50, 50, 50）之间进行多次往复（见图 4-19），这种情形更加明显。在图 4-14～图 4-19 所示的几种条件中，PSOwDOW 执行的效果明显好于标准 PSO 方法，PSOwDOW 方法不仅能够适应优化函数的这种动态改变，并且能够达到一定的精度，即使在步长和频率都较大的条件下（见图 4-16），误差值仍然保持在一定的范围内，这说明 PSOwDOW 方法对优化函数的动态改变有很强的适应性，有能力跟踪最优解的动态改变。

考查 PSO 和 PSOwDOW 在环形模型中的运行情况（见图 4-20～图 4-23），取 T=25，从图中显然可以看出，PSOwDOW 得到的结果明显好于 PSO，在环形模型中，全局最优

解在一个封闭的环形轨迹上移动，图 4-20 中 s=0.01、t=5，这种情况下 PSOwDOW 方法产生的误差小于 10e-5，由于步长较小，PSO 方法产生的误差也基本稳定。放大步长，令 s=0.1（见图 4-21），此时，从图中可以清晰地看出，PSO 方法的误差曲线呈周期变化，这种误差曲线的周期变化和图 4-16 有些相似，也说明在这种条件下，PSO 方法没有能力跟踪最优解的变化。

在随机模型中（见图 4-22～图 4-23），当 s=0.01 时（见图 4-22），全局最优解偏移步长较小，且在（0,0,0）附近随机波动，当算法执行到一定程度时，这种小的波动会发生在 PSO 方法群体的搜索范围内，因此，误差曲线变化逐渐收敛。增大步长，令 s=1（见图 4-23），这时随机扰动的步长变大，全局最优点会跳出群体的搜索范围，PSO 方法无法立刻响应这种变化，造成误差随之增大，跟踪失败。

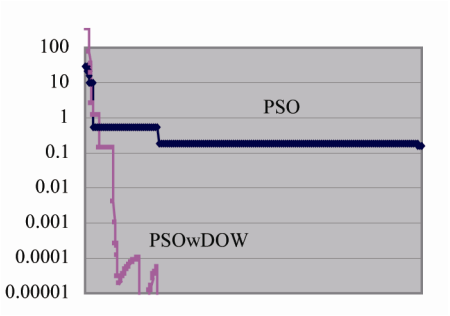


图 4-20 环形模型s=0.01，t=5

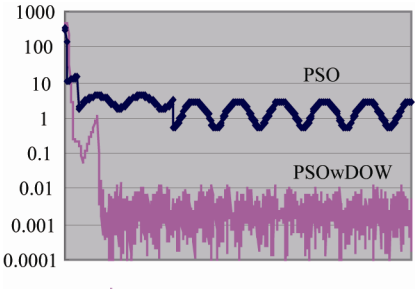


图 4 -21 环形模型s=0.1，t=5

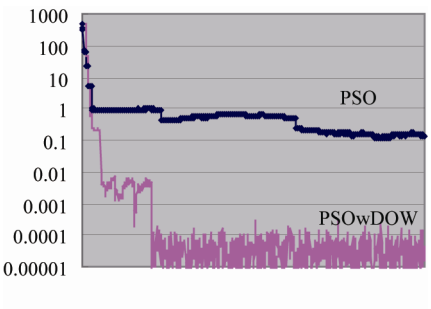


图 4-22 随机模型s=0.01，t=5

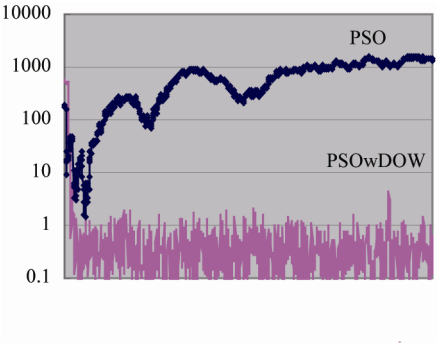


图 4-23 随机模型s=1，t=5

从整体上看，PSO 方法对动态优化环境的适应性并不是很强，只有当扰动频率较低且扰动步长较小的条件下才能对最优解的动态改变进行有效的跟踪。相比之下，

PSOwDOW 更能适应动态函数的这种变化，并且能够达到很高的精度，即使在扰动的频率和步长都很大的条件下（见图 4-17），虽然精度受到一定的影响，但误差仍然保持在一定的范围内。文献[92]对 PSO 方法进行了某些改进，提出了 APSO 方法，并在类似的环境下进行了测试，通过比较可以看出，PSOwDOW 所得到的结果好于 APSO，详见文献[92]。

下面考查 PSOwSA 在几种动态环境下的执行情况。从执行的过程来看，PSOwSA 跟踪动态最优点的能力与标准 PSO 方法相似。取 $a=0.95$，图 4-24～图 4-26 所示为 PSOwSA 在不同参数条件下的误差曲线。

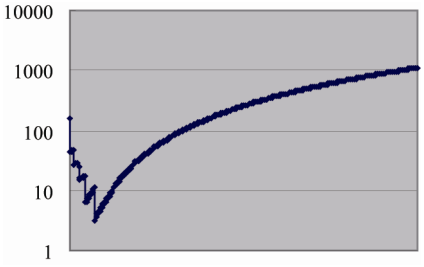


图 4-24　线性模型s=0.1，t=5

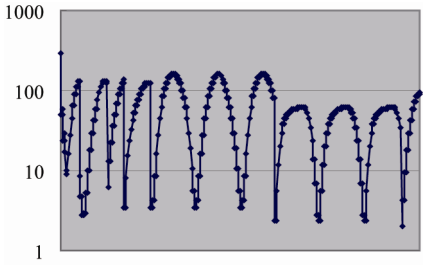


图 4-25　环形模型s=1，t=5

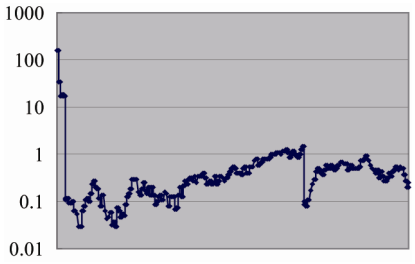


图 4-26　随机模型s=0.1，t=5

从误差曲线上看，PSOwSA 方法在动态优化环境下的表现并不比标准 PSO 好，有的甚至不如标准 PSO。因此，通过实验，可以得出这样的结论：PSOwSA 方法在解决静态优化问题时，所得结果要好于标准 PSO 方法，而对于动态优化而言，PSOwSA 方法产生的误差要大于标准的 PSO 方法。换句话说，PSOwSA 方法和标准 PSO 方法都不适合解决动态优化问题。

标准 PSO 方法总是要设置粒子的参数，使粒子的运行轨迹收敛，这时所有粒子都追随群体最优和个体最优，在空间内进行搜索，粒子在这段时间内，将在以这两点为对

角线的超立方体内呈正态分布。在动态优化环境中，当扰动步长和频率都较小时，问题的最优点在群体的搜索区域中停留时间相对较长，这样就使群体有充分的时间响应环境的这种变化；反之，当扰动步长和频率都较大时，问题的最优点迅速远离群体的搜索区域，算法虽然对群体中的粒子进行了重新评价，但搜索空间并没有因此而改变，所以，误差会不断增大，跟踪失败。

与标准 PSO 相比，PSOwSA 方法的修正策略只能提高标准 PSO 方法的局部搜索能力，以某种概率来修正粒子的下一步位置，削弱了群体的“开拓”能力，当有扰动存在时，最优点逐渐远离群体目前的搜索区域，群体无法及时适应这种状态，进而造成了误差的进一步增大。若调整温度，使之下降得充分缓慢，可以使 PSOwSA 方法适应动态优化环境的能力增强，但是，温度下降得越慢，所需的计算开销就越大，虽然精度会有所提高，但执行效率并没有得到改善。

与上述两种方法相比，PSOwDOW 方法更能适应优化函数的动态改变，原因如下。PSOwDOW 把整个群体分成 3 个独立的子群体：POP_Core、*P*_Near 和 *P*_Far。其中，只有 *P*_Near 中粒子运行策略和标准 PSO 方法的运行策略一致，在这个集合中，设置粒子参数，使 $\max(\|l_3\|,\|l_2\|)<1$，粒子的搜索范围是 Core_Centre 与粒子个体最优 p_i 之间的区域，通常情况下，这一区域被认为是最有希望发现群体最优和个体最优的区域。其他两个集合中的粒子运行方式作如下改进。

① 由本章 4.8 节相关阐述可知：子群体 POP_Core 始终是群体最优粒子的集合，在该集合中采用 EP 策略，以克服标准 PSO 方法局部搜索较差的弱点。通过这种改进，使粒子不断地在群体最优附近探索新的群体最优，从而保证了群体有较强的局部探索能力。

② *P*_Far 中粒子运行轨迹并不收敛，因此，这部分粒子的搜索范围并不局限于以群体最优和个体最优为对角线的立方体中，而是围绕以 p_g 和 p_i 为端点的线段缓慢发散，从而保证了个体的多样性，避免算法“早熟”现象的发生。在动态优化环境中，当动态函数发生变化时，整个群体将进行重新评价，这时，*P*_Far 中的粒子往往更加容易接近

新的全局最优点，成为新的群体最优，整个群体的搜索趋势也随之改变，粒子群将追随新的群体最优在新的空间进行搜索，使群体避免在原来的搜索区域内停滞不前。

这些改进不仅使 PSOwDOW 方法对动态优化环境有很强的适应能力，并且提高了算法的搜索精度。

4.11　小结

本章阐述了粒子群优化方法的相关背景，描述了经典粒子群优化方法的执行过程。从广义上来讲，粒子群优化方法属于演化计算中对群体智能行为的简单模拟。与其他演化计算方法一样，粒子群方法的理论基础还不坚实，文献[85]～[89]等都给出了一些有益的结果，在此基础上，定性地给出了关于粒子运行轨迹的定理，虽然定理中的假设与实际情况存在一定的差异，但仍能够给算法的改进及参数设置提供一些有意义的指导。

一些研究显示[83][84]：PSO 方法具有很强的“开拓”能力，但在最优解附近的“探索”能力却不理想。结合对 PSO 方法的分析，本章提出了模拟退火粒子群优化（PSO with Simulated Annealing，PSOwSA）和有分工策略的粒子群优化（PSO with Division of Work，PSOwDOW）两种不同改进方法，并详细阐述了这两种方法的主要思想。

在 PSOwSA 中，把模拟退火机制引入到 PSO 方法中，当粒子下一步位置比当前位置好时，粒子移动到下一步位置；反之，若下一个位置比当前位置差，则粒子并不直接移动到下一个位置，而是以某一概率进行移动，且通过温度来控制这一概率。若把标准 PSO 看成对鸟类觅食等群体行为的简单模拟，则在 PSOwSA 中的“鸟”更加聪明，它不是盲目地直接扑向下一个位置，而是以某种概率“试探”后再行动。因此，当温度下降得足够慢时，粒子不会轻易地跳出有“希望”的搜索区域，从而增强了粒子的局部搜索能力。

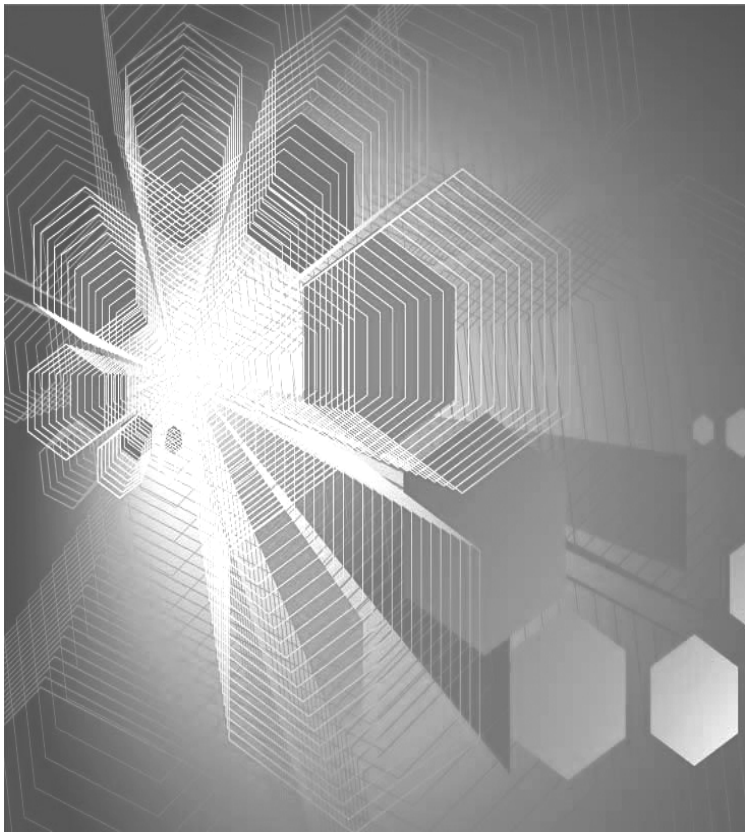
在 PSOwDOW 中，为增强群体的局部搜索能力，把粒子群划分为 3 个互不交叉的子群体并作分工，在执行过程中这 3 个群体不断地发生变化。这种处理方式较好地平衡

了局部“搜索”和“开拓”新领域之间的关系，在计算量没有明显增加的前提下，计算结果却有一定的提高。

对两种改进方法进行测试，测试结果表明：对于静态优化问题，PSOwDOW 和 PSOwSA 两种改进方法都不同程度地克服了传统 PSO 方法的弱点，增强了群体的优化能力，对解决静态优化问题有一定的应用价值。

为进一步考查两种改进算法的性能，本章研究了这两种方法在动态优化环境下的适应性。试验结果表明：传统 PSO 方法和 PSOwSA 方法对动态优化环境的适应性并不是很强，只有当扰动频率较低且扰动步长较小的条件下，才能对最优解的动态改变进行有效的跟踪；而 PSOwDOW 方法对动态优化环境有很强的适应性，能够有效地追踪全局最优点的动态改变，并保持较高的精度。即使最优解扰动频率和扰动步长都很大的条件下，产生的误差依然在某一范围内保持稳定。这表明 PSOwDOW 方法在很大程度上提高了对动态优化环境的适应性，其不仅适合静态优化问题，在解决噪声条件下的复杂非线性优化问题上也有较强的应用价值。

第 5 章　微分演化



对求解优化问题而言，通常要求优化技术能满足以下三点。第一，无论系统初始参数如何设置，方法应当能找到真正的全局最优值；第二，收敛速度要快；第三，方法的控制参数尽可能少，参数敏感性弱。微分演化（Differential Evolution，DE）似乎具有这样的性质，该方法由 Storn 和 Price 在 1995 年提出[95]，它与 PSO 方法几乎产生于同一时间，这种方法不仅简单，而且在对各种测试问题的实验中表现优异。DE 的基本思想如下：利用两个随机选取的矢量参数的差向量作为第三个矢量参数的随机变化源。微分演化区别于遗传算法、进化策略、进化规划、PSO 等方法，从表面上看它与 PSO 方法有相似之处，同时也具备遗传类算法的一些特征，DE 利用两个随机选取的矢量参数的差向量作为第三个矢量参数的随机变化源，而 PSO 方法利用的是个体最优和群体最优来确定下一步的位置。国外有许多关于 DE 方法的研究[95]~[101]，本章对 DE 方法进行了详尽的阐述，对先前的一些研究进行了归纳，同时，介绍了一种 DE 和 PSO 的混合方法——DEPSO。

5.1 微分演化方法描述

DE 是一种新颖的并行搜索方法。首先，在搜索空间内，随机产生初始群体。通过将群体中两个成员间的差向量增加到第三个成员的方法来生成新的个体。如果新个体的适应度值更好，那么，新产生的个体将代替原个体。具体过程如算法 5-1 所述。

算法中 CR、F 为控制参数，其中，CR 被称为交叉常数（Crossover Constant），F 被称为缩放因子（Scaling Factor），$j_{\text{rand}} = \text{random}(1, D)$ 为[1,D]上的随机整数，random(0,1)为[0,1]上满足正态分布的随机数。从算法 5-1 中可以看出，有 3 种运算贯穿着整个算法的执行过程：变异（Mutation）、交叉（Crossover）和选择（Selection）。

1．**变异**（Mutation）

对每个目标向量 $\bar{\boldsymbol{X}}_i(t)$，根据公式

$$\bar{\boldsymbol{V}}_i(t+1) = \bar{\boldsymbol{X}}_{r_3}(t) + F(\bar{\boldsymbol{X}}_{r_1}(t) - \bar{\boldsymbol{X}}_{r_2}(t)) \qquad (5\text{-}1)$$

生成变异向量，其中，$\bar{\boldsymbol{X}}_{r_1}(t)$，$\bar{\boldsymbol{X}}_{r_2}(t)$，$\bar{\boldsymbol{X}}_{r_3}(t)$ 为群体中随机选择的 3 个个体，i 和 r_1、r_2、r_3 之间必须是不同的。因此，一个群体中个体的数量必须多于 4 个。F 是一个[0,2]上的实型常量因子，用于控制差向量($\bar{\boldsymbol{X}}_{r_1}(t)-\bar{\boldsymbol{X}}_{r_2}(t)$) 的影响大小。

	算法 5-1　微分演化方法
0	Algorithm: Differential Evolution, DE
1	Let t=0,Create a random initial population $\{x_i(t)\}$，i=1, $\mathbf{L}$,population_size
2	Evaluate $f(x(t))$, i=1, $\mathbf{L}$, population_size
3	For t=1 to MAX GENERATIONS
3.1	For i=1 to population_size
3.1.1	Select randomly $r_1 \neq r_2 \neq r_3$ in population and Let j_{rand}=random(1,D)
3.1.2	For j=1 to D Do
3.1.2.1	If (random(0,1)<CR or $j = j_{rand}$) Then
3.1.2.1.1	$u_j^i(t+1) = x_j^{r_1}(t) + F(x_j^{r_2}(t) - x_j^{r_3}(t))$
3.1.2.2	Else
3.1.2.2.1	$u_j^i(t+1) = x_j^i(t)$
	Endif
3.1.3	End for
3.1.4	If $f(u^i(t+1)) \leqslant f(x^i(t))$ Then
3.1.4.1	$x^i(t+1) = u^i(t+1)$
3.1.5	Else
3.1.5.1	$x^i(t+1) = x^i(t)$
	Endif
3.2	End for
3.3	$t = t + 1$
4	End for

注意，父个体 $\bar{\boldsymbol{X}}_{r_1}(t)$、$\bar{\boldsymbol{X}}_{r_2}(t)$ 之间的差向量越小，扰动就越小，这意味着如果群体靠近优化值，步长会自动减小。这与标准演化策略中的自动步长控制类似。

为了更直观地反映 DE 方法，图 5-1 描述了个体的上述修正策略。

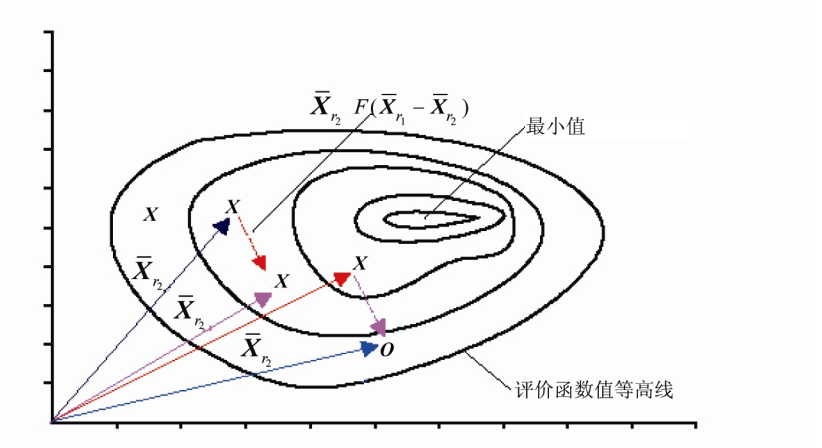


图 5-1　新向量的生成

2．交叉（Crossover）

目标向量与变异向量相结合，生成新的向量

$$u_i(t+1)=(u_{i1}(t+1),u_{i2}(t+1),\mathbf{L},u_{iD}(t+1))$$

其中，D 为问题空间的维度，

$$u_{ij}(t+1)=\begin{cases}v_{ij}(t+1) & ,\ \text{random}(0,1)<\text{CR}\ \ \text{or}\ \ j=j_{\text{rand}}\\ x_{ij}(t) & ,\ \text{random}(0,1)>\text{CR}\ \ \text{and}\ \ j\neq j_{\text{rand}}\end{cases}\qquad(5\text{-}2)$$

其中，$i=1,\mathbf{L},\text{population\_size}$；$j=1,\mathbf{L},D$。

这里，CR 是交叉常数，在[0,1]上，CR 值越大，发生交叉的可能就越大。CR=0 表示没有交叉。j_{rand} 是[1,D]上随机选择的整数，它保证 $u_i(t+1)$ 至少要从 $v_i(t+1)$ 中获得一个元素。否则，就不会有新的向量生成，群体也不会变化。图 5-2 所示为 DE 信息交换示意。

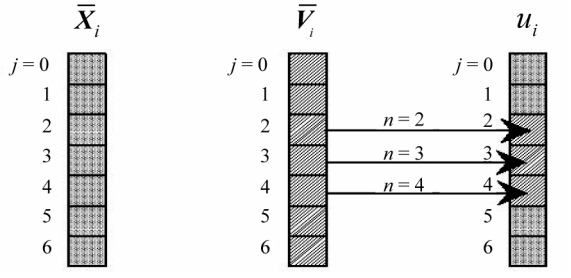


图 5-2　DE信息交换示意

3．**选择**（Selection）

在标准 DE 方法中，使用的是一种“贪婪”选择模式：当且仅当新个体的评价函数值更好时，才被保留到下一代群体中。否则，父个体仍然保留在群体中，再一次作为下一代的父向量。

DE 方法存在一些其他变化，可以用符号 DE/***X***/***Y***/***Z*** 区分，其中，***X*** 是确定将要变化的向量，***Y*** 是需要使用差向量的个数，***Z*** 表示交叉模式。例如，***X*** 可以是 Rand（随机在群体选择个体）或者 Best（选择当前群体中的最佳个体），***Y*** 是使用差向量的个数，当前变量 ***Z*** 值是 bin，表示交叉操作的概率分布满足二项式试验。使用这种符号规则，基本的 DE 方法可以描述为：DE/Rand/1/bin。

若将式（5-1）修改成下式：

$$\bar{\boldsymbol{V}}_i(t+1)=\bar{\boldsymbol{X}}_i(t)+l\,(\bar{\boldsymbol{X}}_{\text{Best}}(t)-\bar{\boldsymbol{X}}_i(t))+F(\bar{\boldsymbol{X}}_{r_2}(t)-\bar{\boldsymbol{X}}_{r_3}(t)) \qquad (5\text{-}3)$$

其中，$\bar{\boldsymbol{X}}_{\text{Best}}(t)$ 表示第 *t* 代群体中最好的个体，采用上述符号规则表示为：DE/Best/2/bin。

研究表明，DE/Bset/2/bin 看起来似乎比 DE/Rand/1/bin 要好。DE/Best/2/bin 的修正向量中增加了当前群体的最佳向量，这表明群体有向群体最优点移动的趋势。因此，当群体成员收敛到一个局部最优解时，从局部最优中逃脱的机会也就越小。

由于对 DE/Best/2/bin 来说，两个差分矢量都增加到了目标矢量上，它的缩放因子 ***F*** 通常应该小于 DE/Rand/1/bin 的。一个具有 6 个个体的群体，对 DE/Best/2/bin 就可能有 360 个不同的扰动矢量，而对 DE/Rand/1/bin 只有 30 个。这表明使用 DE/Best/2/bin 能比 DE/Rand/1/bin 生成更多的不同的试验矢量。

5.2　DE 参数的设置

在 DE 方法中，有 3 个策略参数：population_size 是群体中成员的个数；***F*** 是差向量的缩放因子；CR 是交叉常数。实验表明，DE 方法的优化能力和收敛速度与控制变量 population_size、***F*** 和 CR 的选择紧密相关[96]。

1．群体规模 population_size

经验显示，合理的群体规模大小应介于 3*D* 和 8*D* 之间。对于 DE/Rand/1/bin，population_size 必须至少为 4*D*，而对于 DE/Best/2/bin，群体规模至少为 5*D*，以此保证 DE 有足够变化的差向量。群体规模越大，搜索能力就越强。然而，大的群体规模也需要大量的个体评价运算。

2．缩放因子 *F*

一般来说，*F* 应该不小于某一特定值，这样能够有效地避免算法过早收敛。较大的 *F* 增加了从局部最优逃脱的可能性。然而，若 *F*>1；算法的收敛速度会明显降低。对一个群体来说，当扰动大于两个成员间的距离时，收敛会更困难。经验表明，放大因子的一个比较好的初始值是 *F*=0.6。

3．交叉常数 CR

较大的 CR 常常会加速收敛。一般情况下，交叉常量较好的选择是[0.3,0.9]。

5.3 算法仿真

选择附录 A 中的部分测试问题对 DE 方法进行测试，实验结果展示了该方法强大的搜索能力。

5.3.1 低维条件下的仿真结果

f_1～f_{23} 的详细介绍已在附录 A 中给出。DE 的 3 个参数：群体规模（population_size），交叉常数（CR）和缩放因子（*F*）在所有实验中被设为 population_size =100，CR=0.9，*F*=0.6。所有问题都运行 30 次，如表 5-1 所示为 DE 方法得到最优解的平均值。

对于 f_1～ f_4，DE 方法表现出很好的性能，快速地收敛于最优解；对于函 f_5，DE 在 300000 次迭代后，开始以指数速度在最优解周围调整。最后收敛于最优解。函数 f_6 是阶梯函数，DE 能够容易找到其全局最优，f_7 是一个噪声问题。DE 方法收敛速度情况，略

差于其他算法。f_8～ f_{13}是多峰函数，DE 在所有情况下都找到了全局最优解。对于函数 f_{14}，DE 虽然没有得到全局最优解，但在所有的运算中都非常接近全局最优解。对于函数 f_{15}，DE 向最优解附近的值收敛非常快，但是之后的搜索却一直处于停滞状态。函数 f_{16}～f_{23}相对简单，DE 找到了所有条件下的最优解，又一次表现了它优异的性能。

表 5-1　维度为 30 时，DE 运行 30 次得到各 Benchmark 问题最优解的平均值

函数	最优解	DE 得到的最优解	函数	最优解	DE 得到的最优解
f_1	0	<10e-8	f_{13}	-1.1428	-1.1428244
f_2	0	<10e-8	f_{14}	0.998	0.9980039
f_3	0	<10e-8	f_{15}	0.0003075	4.1736828e-4
f_4	0	3.8502177e-8	f_{16}	-1.0316	-1.0316285
f_5	0	<10e-8	f_{17}	0.398	0.39788735
f_6	0	<10e-8	f_{18}	3	3
f_7	0	4.9390831e-3	f_{19}	—	—
f_8	12569.5	-1.2569481e+4	f_{20}	—	—
f_9	0	<10e-8	f_{21}	-10.2	-10.153201
f_{10}	0	-1.1901591e-15	f_{22}	-10.4	-10.402943
f_{11}	0	<10e-8	f_{23}	-10.5	-10.536412
f_{12}	0	<10e-8			

5.3.2　高维条件下的仿真结果

当搜索空间维度为 100 时，f_1的情况与在 30 维时结果相似，DE 以指数水平迅速收敛到全局最优解；对于 f_2～ f_4，DE 收敛速度受到一定的影响，但仍然收敛到全局最优值；对于相对困难的函数 f_5，DE 方法在 350 万次迭代后，找到了最优解；对于其他函数，DE 的表现和 30 维时类似，只是迭代次数变得更大。

关于 DE 在搜索空间维度是 100 时的具体执行情况，在表 5-2 中进行了具体说明。在本书前面的章节中，对高维优化问题也进行了研究，其中包括对搜索空间是 100 时，一些算法执行情况的研究，但是试验条件与本节中的条件不尽相同，在后面的研究中，还要将 DE 方法和本书提出的其他方法进行细致的比较。

表 5-2　当搜索空间为 100 时，DE 运行 30 次得到各 Benchmark 问题最优解的平均值

函数	最优解	DE 得到的最优解	函数	最优解	DE 得到的最优解
f_1	0	<10e-8	f_8	41898.3	-4.1898293+4
f_2	0	<10e-8	f_9	0	<10e-8
f_3	0	<10e-8	f_{10}	0	<10e-8
f_4	0	<10e-8	f_{11}	0	<10e-8
f_5	0	<10e-8	f_{12}	0	<10e-8
f_6	0	<10e-8	f_{13}	-1.1428	-1.1428244
f_7	0	7.6640871e-3			

总体上，DE 是一种非常出色的优化算法。 对大多数问题它都找到了问题的全局最优解。遗憾的是：对于问题 f_7，DE 的收敛速度相对较慢，显然，DE 在这个问题上遇到了困难；在对 f_{15} 的实验过程中，DE 常常会停滞在一个不太理想的局部最优上。虽然存在一些问题，但是 DE 改善性能的潜力依然很大。DE 的稳健性很强，它能在很多试验中不断产出相同的结果。与 PSO 方法相比，算法对参数变化的敏感性相对较弱，当改变问题时，一般不需要改变参数以维持最佳性能。在试验过程中，DE 显示了很好的调整性能，但是对于一些特殊问题，如噪声问题，DE 与其他方法相比并不成功[96]，目前还不能确定为什么 DE 没能调整好以适应这些特殊的问题。

5.4　微分演化粒子群优化

前几章对粒子群优化（Particle Swarm Optimization，PSO）进行了详细的阐述，DE 方法与之有一定相似之处。在 PSO 方法中，在粒子根据自身经验和群体经验修正自身位置，它依赖群体中其他个体的经验。这一点，DE 方法和 PSO 有相似之处，DE 也是先根据 3 个不同个体生成一个新的个体，与 PSO 不同的是：DE 方法并不用这个新个体直接取代原来的父个体，而是按某种概率与新生成的个体进行信息交换。这一点又与遗传算法有些相似，最后采用“贪婪”的选择模式，淘汰适应度较差的个体。由于参与交换信息的新个体由 3 个不同的个体生成，与要修正的个体没有什么联系，因此，DE 的

这种个体修正策略较大程度地保证了群体的多样性。一般来说，种群的多样性是保证算法不至于过早收敛的基础。

文献[102]提出了一种使用微分演化操作的粒子群方法（Differential Evolutionary Particle Swarm Optimization，DEPSO），这种方法把 DE 方法操作引入到 PSO 方法中去。下面将加以阐述。

在 DEPSO 中，把 DE 中的操作施加在 p_i 上，使之具有一个跟踪点 $T_i = p_i$，其中，对于第 d 维有

$$\text{If } (\text{random}(0,1) < \text{CR} \quad \text{or} \quad j = j_{\text{rand}}) \text{ Then } \quad T_{id} = p_{gd} + d_{2,d} \tag{5-4}$$

其中，j_{rand} 是一个[1,D]上的随机整数，D 是问题搜索空间的维度。式（5-4）保证了至少有一维会发生变化。CR 是交叉常数，如上文所定义；$\vec{d}_2$ 是当 N=2 时的通用偏移向量，由下式定义:

$$\vec{d}_N = \left(\Sigma_1^N \vec{D}\right) / N \tag{5-5}$$

其中，$\vec{D}$ 是基本偏移向量，表示从包含所有的 pbest 的公共点集合中随机选择两个元素的差向量，N 是所包含的 $\vec{D}$ 的数量。对于第 d 维来说，满足

$$\vec{D}_d = \vec{p}_{A,d} - \vec{p}_{B,d} \tag{5-6}$$

其中，$\vec{p}_{A,d}$ 和 $\vec{p}_{B,d}$ 是在 pbest 集合中随机选择的两个点。

现在来试验分析 f 或 $\vec{d}_N$，假设问题的搜索空间是 1，这样的限制并不失其一般性。图 5-3 所示分别为对 $\vec{d}_1$ 和 $\vec{d}_2$ 在 100 万次反复测试中偏移概率分布的柱状图[102]。每个用来计算 $\vec{D}$ 的维元素 p 是从[-1, 1]上随机挑出的实值。可以看出，$\vec{d}_1$ 是一个三角形分布，而 $\vec{d}_2$ 是一个“钟形”分布，文献[103]指出后者对问题解决效果更好。因此，在式（5-4）中选择 N=2。

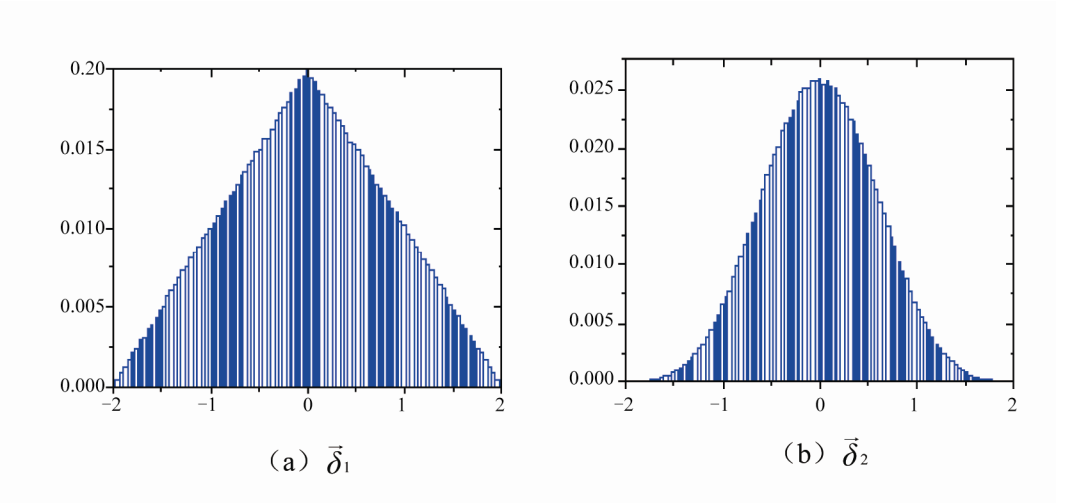


图 5-3 $\vec{d}_1$ 和 $\vec{d}_2$ 在 100 万次反复中测试中偏移概率分布图

在这里，变异基于 p_g，为个体提供了部分群体经验，这会加快算法的收敛速度。学习速率由 CR 决定，如果 CR=1，那么式（5-4）就变成关于 p_g 的一个满足了图 5-3（b）中偏移分布的一种演化策略（ES），如果 CR<1，它可能会保留一些 p_i 的信息，这一策略使算法不至于收敛到 p_g 附近。在 DEPSO 中，$c_1 = c_2 = 2$，$w = 2$，如果没有特别说明，每个测试实例运行 T=2000 次，所有的实例每个运行 100 次[102]。如果新个体在某一维度上超出了规定的范围，则采用下式进行修正：

$$\text{If} \quad (x_{id} < l_d) \text{ Then } \quad x_{id} = u_d - ((l_d - x_{id}) \bmod s_d) \qquad (5\text{-}7)$$

$$\text{If} \quad (x_{id} > u_d) \text{ Then } \quad x_{id} = l_d + ((x_{id} - u_d) \bmod s_d) \qquad (5\text{-}8)$$

其中，mod 是模运算，l_d 和 u_d 分别是搜索空间在维度 d 上的下界值和上界值，$s_d = |u_d - l_d|$。文献[102]采用 Rosenbrock 函数 f_5、广义 Rastrigrin 函数 f_9 和广义 Griewank 函数 f_{11} 对 DEPSO 进行了测试，具体试验结果参见文献[102]中的 Table 1。该表显示出 DEPSO 比所有的传统 PSO 性能更好，尤其对 Rastrigrin 和 Griewank 函数，但是从这个表的结果看，DEPSO 似乎还比不上文献[96]提供的结果，这也许是参数设置上的不同造成的。DEPSO 把 DE 算子引入到 PSO 方法中，使 PSO 的性能不会对策略参数的选择过于敏感，其思想有一定的借鉴意义。

5.5　用 DE 确定 PSO 的最佳参数

PSO 方法通常用来解决无约束优化问题。其中，惯性参数 w 及感知系数 c_1、c_2 决定了 PSO 方法的性能，同时，PSO 方法对参数有很强的敏感性，因此，如何确定合适的参数是 PSO 方法的关键。这些理论研究用以指导 PSO 方法的参数设置，但是，这些研究多基于一种理想状态，与实际计算存在一定的差异：在实际计算中，粒子往往在一定空间（如 N 维空间内的超立方体）内进行搜索，若粒子在某一维上超出了搜索范围，则把粒子在该维度上的值设置成搜索区间的上界或下界。这种情况在 PSO 的执行过程中经常会发生，而且是随机的，无法进行判断和估计。这就造成了在实际计算中粒子运行轨迹与理论分析中结论的差异。因此，用这些理论来指导参数设置是不完全可靠的。通过大量的实际计算和实验，为 PSO 方法的参数设置积累了一定经验，这些经验是非常有意义的。但是，由于问题的复杂性，对于一个复杂的新问题，这些经验是否有效依然是一个值得研究的问题。调整参数直至运行结果满意固然是一种方法，但这种作法效率非常低，并且需要凭借应用者的直觉和经验，给 PSO 方法的应用带来难度。本节介绍通过一种微分演化技术确定 PSO 参数的策略，如算法 5-2 所示。

算法 5-2　用 DE 确定 PSO 的最佳参数

0　Algorithm: Algorithm: Determined parameters of PSO using DE

1　在合适的区间内，随机产生由参数向量构成的群体 $\mathrm{POP}=\{\boldsymbol{P}_1,\boldsymbol{P}_2,\mathbf{L},\boldsymbol{P}_k\}$，其中，$\boldsymbol{P}_i=(w_i,c_{1i},c_{2i})$，$i=1,2,\mathbf{L},k$；对于每个 $\boldsymbol{P}_i$，用下式来评价 PSO 方法的执行性能：

$$p_i = f\_\mathrm{pso}(\boldsymbol{P}_i)=\frac{1}{N}\sum_{m=1}^{N}\mathrm{optimum}(\boldsymbol{P}_i, f(x)) \qquad (5\text{-}9)$$

其中，$\mathrm{optimum}(\boldsymbol{P}_i, f(x))$ 表示当参数设置为 $\boldsymbol{P}_i$ 时，用 PSO 方法求得的 $f(x)$ 的最优解；

2　对 $\forall \boldsymbol{P}_i \in \mathrm{POP}$，按 DE 方法的修正策略对其进行修正，生成一个新的向量 $\boldsymbol{P}_i'$，使群体中个体的数量增加一倍；

3　按式（5-9）对 POP 中的所有参数向量进行评价，按某种策略选择出 k 个优秀个体重新组成下一代群体；

4　重复执行步骤 Step2、Step3 直至满足终止条件。

如图 5-4 和图 5-5 所示分别为 $p=\frac{1}{10}\sum_{m=1}^{10}\mathrm{optimum}((1,c_1,c_2),f_1)$ 和 $p=\frac{1}{10}\sum_{m=1}^{10}\mathrm{optimum}((0.7,c_1,c_2),f_1)$ 的曲面（事实上，这两个曲面的全局最优点就是当 w=1 和 w=0.7 时，PSO

方法的最优参数，这里的目的只是初步确定感知系数 c_1、c_2 的范围，因此，这对两个曲面的精度要求并不高，若要得到高精度的参数曲面，计算量相当庞大。从这两个曲面上分析，确定 c_1、c_2 的值在[1,2.5]上。

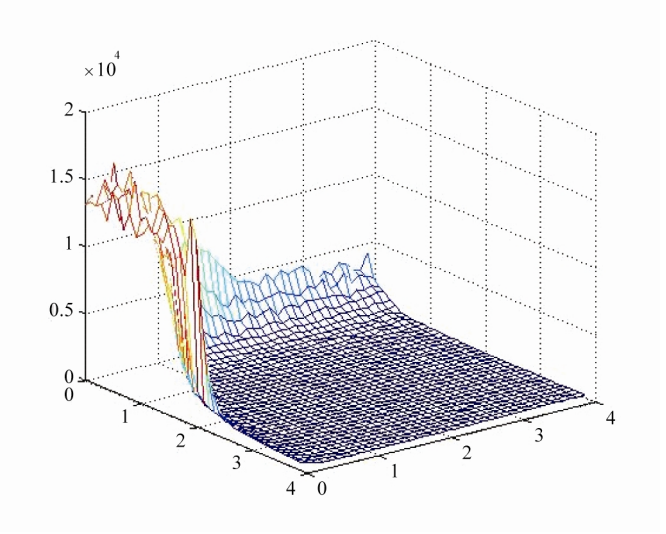


图 5-4　w=1，感知系数c_1、c_2与PSO性能曲面

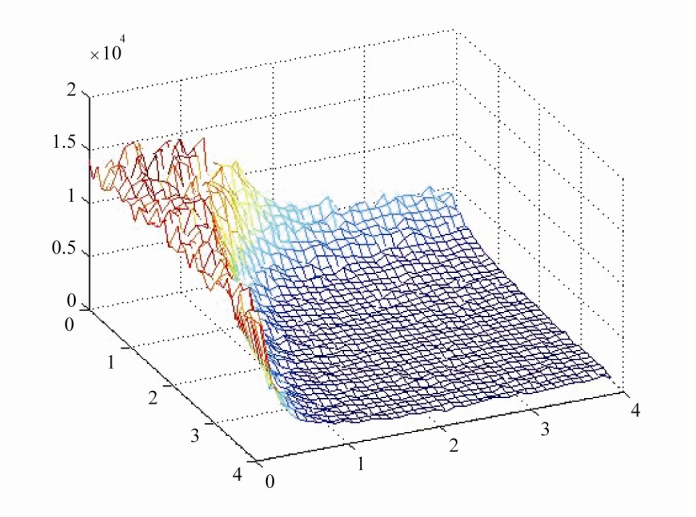


图 5-5　w=0.7，感知系数c_1、c_2与PSO性能曲面

用同样的方法，固定 c_1=1.4，c_2=1.4，考查 w 对 PSO 的影响，图 5-6 所示为函数 $p=\frac{1}{10}\sum_{m=1}^{10}\text{optimum}((w,1.4,1.4),f_1)$ 的曲线，根据该曲线确定 $w\in[0.6,1.15]$。

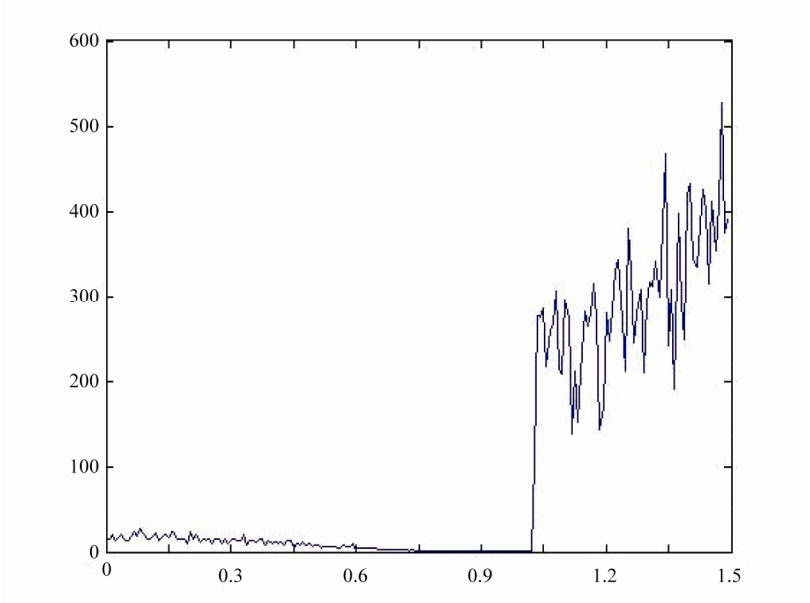


图 5-6　感知系数c_1、c_2一定，w与PSO性能曲线

由此可知，DE 方法中个体 $\boldsymbol{P}_i=(w_i,c_{1i},c_{2i})$（$i=1,2,\mathbf{L},k$）的搜索空间为 $S=[0.6,1.15]\times[1,2.5]\times[1,2.5]$，为 DE 方法设置合理的参数，按照本节所述过程进行实验，得到 PSO 方法关于上述问题的最优参数如表 5-3 所示。

表 5-3　最优参数表

No.	Func	w	c_1	c_2
1	f_1	0.7316	1.3921	1.5113
2	f_2	0.7841	1.4018	1.4271
3	f_3	0.8004	1.3258	1.4785
4	f_4	0.7214	1.4224	1.4009

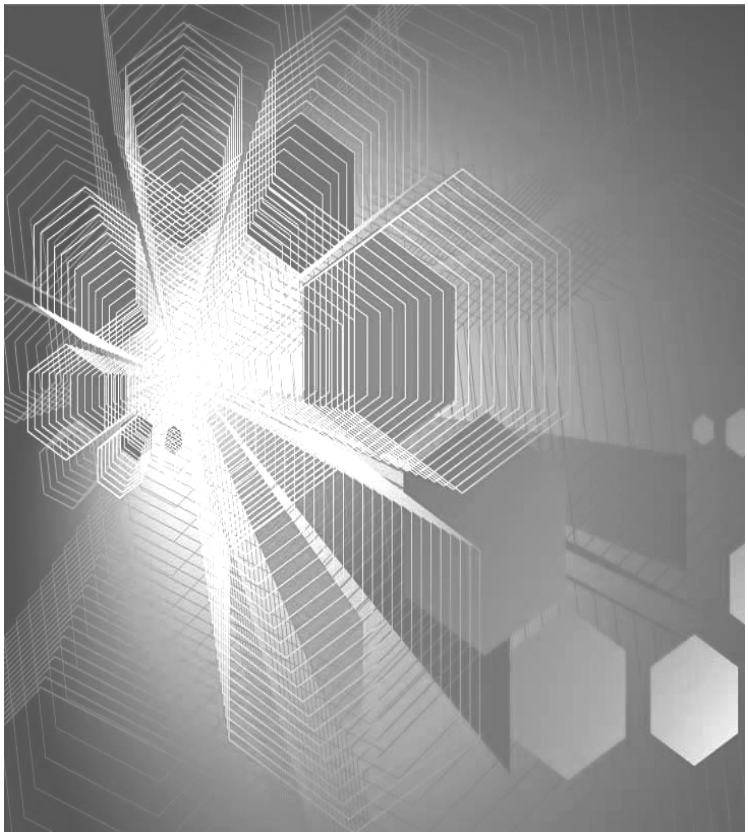
上述结果与经验值接近，但并不一致，这种差异是正常的。同时，不同优化问题所采用的参数也不应该相同。当遇到一个复杂优化问题，经验的参数设置未必可靠时，则可以应用上述策略，为 PSO 方法选择针对该问题的合理参数。可以看出，本书提出的策略是不依赖个人经验的，并且在求得最优参数的同时，优化问题的最优解也同时被给出。

5.6 小结

DE 方法简单、收敛快、性能突出。它只有少量参数需要设置，而且同一设置能应用到不同问题，表现出了它在求解优化问题中的价值，当面临解决一个新的优化问题时，DE 可以被认为是优秀的选择。在本书的研究中，对大多数 Benchmark 问题，DE 都得到了较好的结果。

DE 方法提出的时间较短，与 PSO 方法产生于同一个年代，但算法对控制参数的敏感性不像 PSO 那么强烈。尽管 DE 具有这些令人期望的特性，但是 DE 仍然是一个刚刚起步的新方法，并且有可能继续完善。进一步的研究包括：数学上的收敛证明，就像其他算法的理论证明一样。系统地分析 DE 为什么收敛如此好将会是一个研究热点。最后，很重要的一点是，仍然需要通过实际应用来了解如何选择 DE 的控制参数，这一点往往需要通过一定的理论分析和大量的实际试验来获得。

第 6 章　文化算法



之前的章节所讨论的问题均是无约束优化问题，本章讨论采用文化算法求解约束优化问题的过程。约束优化问题（Constrained Optimization Problem）可以描述为

$$\min_{x\in S} f(x)$$

满足

$$s_i(x)\geqslant 0 \quad i=1,2,\mathbf{L},m$$

$$h_j(x)=0 \quad j=1,2,\mathbf{L},l\ (l\leqslant n)$$

其中，S 是搜索域，f 是被优化函数，s_i 、h_j 是约束条件集合。

目前，还没有一种现成的方法来确定上述问题的全局最优解。只有当目标函数 $f(\bullet)$ 和约束 s 、h 满足某种性质时，全局最优解才可能被找到。先前介绍的几种方法都是针对无约束最优化问题的，本节将主要讨论几种处理约束优化问题的方法。其中，大多数是最近推出的新方法。几年前，Richardson 等[104]认为，应用遗传算法到约束优化问题的尝试应该遵从如下两个原则：

① 对遗传算子的修正；

② 惩罚不满足所有约束的个体。

随着研究的不断深入，人们发现即使是罚函数法，也是由多种方法组成的，这些方法在罚函数设计和应用于不可行解的许多细节上是不相同的。因此，目前的一些研究已不仅限于这两个原则。一些新策略被提出，主要包括：

① 维持群体中个体的可行性；

② 使用特殊的算子或编码方式；

③ 可行个体优先原则（任何一个可行个体比所有不可行个体都要“好”）；

④ 以一个特定的线性次序来考查约束、修补或惩罚不可行个体等。

本章将对这些策略加以阐述和讨论。同时，介绍一种解决约束优化问题的演化算法——文化算法（Culture Algorithms，CA）。文化算法是基于这样一个理念：它在文化和群体两个空间上同时演化，在解决约束优化问题时，把约束描述成信仰空间（Belief

Space）中的知识，这些知识通常被称为文化（Culture），并且根据它来指导群体空间的搜索。对一些测试函数的数值试验证实了文化方法对约束优化问题的有效性和通用性。

6.1　约束的处理

6.1.1　可行解和不可行解

不连续域中的约束问题较早就被公认了。背包问题、集合覆盖问题、任何类型的日程表和时间表问题都是受约束的。有一些启发式方法用于处理约束，但是，这些方法都没有进行过系统的研究。

在约束优化问题中，不可行个体的处理是非常重要的，在演化计算方法中，评价函数用于连接问题和算法，评价函数对群体中的个体进行打分；较好的个体有更多的机会生存和再生。因此，有必要以一种“完美的方式”定义能赋予问题以特征的评价函数。特别是，处理可行个体和不可行个体时应该十分小心：一个群体常常包含不可行个体，但是，我们的目标是搜索一个可行的最优解。对可行和不可行个体，找到一个合适的评价方法是非常重要的，它直接影响算法的成败。

一般来说，在无约束优化中，总是假设搜索空间 S 是 N 维空间上的一个超立方体，而在约束优化中，搜索空间 S 被各类约束分割成彼此分开的可行搜索空间 F 和不可行搜索空间 U，如图 6-1 所示。

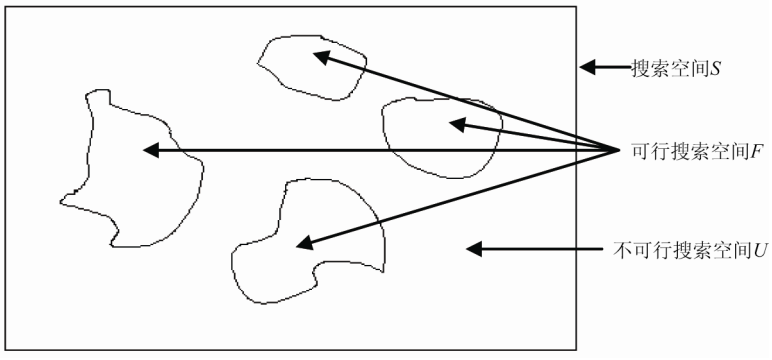


图 6-1　约束问题的搜索空间

对这些子空间没有做任何假定，它们不必是凸集，也不必是相连接的。如图 6-1 中的例子，其中搜索空间的可行部分 F 由 4 个分立的子集构成。在搜索过程中，不得不处理各种可行和不可行个体。如图 6-2 所示，在演化过程中一个群体可能包含可行的个体（*b, c, d, e, i, j, k, p*）和不可行的个体（*a, f, g, h, l, m, n, o*），而全局最优解由“*x*”标记。

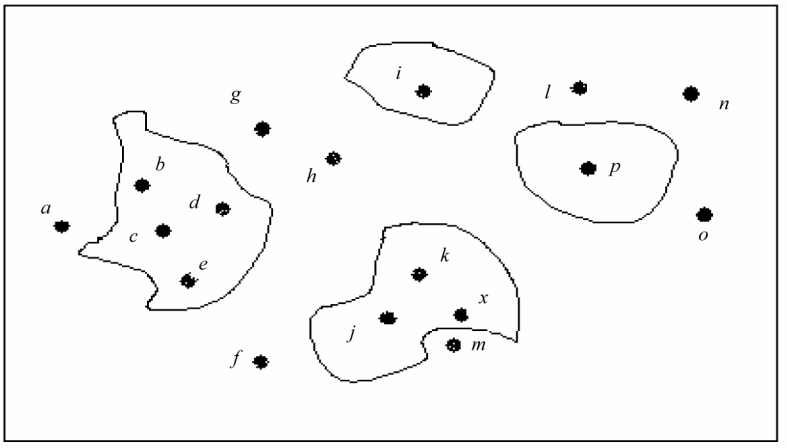


图 6-2　搜索空间和搜索空间上的群体

群体中可行和不可行个体的存在，影响着演化算法的其他部分。例如，是否“精英（Elite）”选择方法应该考虑保存最好的可行个体，或者保存只是总体上的最好个体呢? 再者，某些算子可能只应该用于可行个体。如何评价群体中的个体的问题不是一个小问题。一般情况下，不得不设计两个评价函数 eval_f 和 eval_u，分别针对可行和不可行域。在这方面有许多重要的问题需要在设计算法时具体考虑。

6.1.2　可行个体评价函数 eval_f 的设计

可行个体评价函数 eval_f 的设计通常是最容易的问题，在多数数值优化问题及多数运筹学问题中（背包问题、货郎担问题、集合覆盖问题等），对可行解的评价函数是给定的。但是对某些问题，评价函数的选择仍然是很粗糙的。例如，构造一个演化程序来控制移动式机器人时，就有必要评价机器人的路径。图 6-3 所示，路径#1 和路径#2 哪个路径有更好的评价还不能确定，因为要考虑它们总的距离长短、障碍的大小和光滑性等各种因素。虽然路径#1 更短，但是路径#2 更光滑。对这类问题，有必要将一些启发

式方法引入到评价函数中。这会给评价函数的设计造成一定的难度，往往要具体问题具体对待。

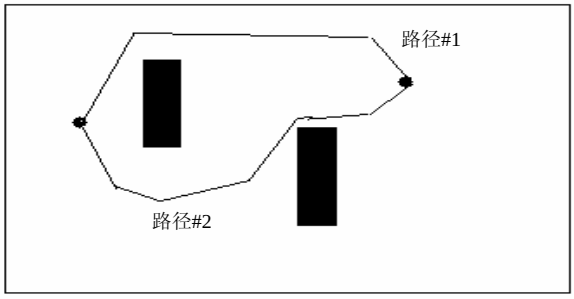


图 6-3　可行路径的比较

对许多设计问题也存在这样的情况，并没有比较两个可行个体的明确的公式。很明显，一些与问题有关的启发式方法在这种情况下是必要的，它将提供对一个可行个体 x 的数值度量。

6.1.3　不可行个体的处理

1．评价函数 eval_u 的设计

与可行个体评价函数的设计相比，eval_u 的设计相对复杂。可以通过排除不可行个体来避免它。有时可以通过函数 eval_f 来处理不可行个体，即 $\mathrm{eval}_u(x)=\mathrm{eval}_f(x)+Q(x)$，其中，$Q(x)$ 表示的是对不可行个体 x 的惩罚，或者对这样个体的一个修补费用。

另一个选择是设计独立的两个评价函数 $\mathrm{eval}_u(x)$ 和 $\mathrm{eval}_f(x)$，但是在这种情况下不得不建立一些这两个函数之间的关系。不可行个体是很难评价的，应该定义一种规则，在这种规则下，构造一种对不可行个体的排序关系。以图 6-4 为例，定义违背约束少的路径好于违背约束多的路径，在这种关系下，路径#1 好于路径#2。这只是一个简单的例子，$\mathrm{eval}_u(x)$ 的设计往往是与问题有关的，需要针对问题具体设计。

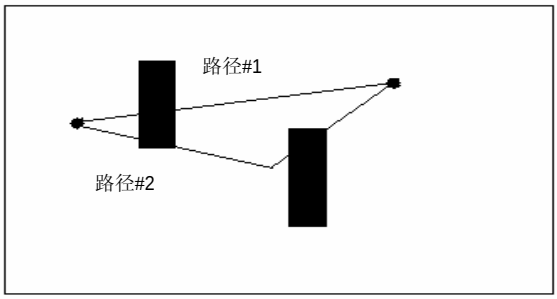


图 6-4　不可行路径

2．函数 $\mathrm{eval}_u(x)$和函数 $\mathrm{eval}_f(x)$之间的关系

假定在群体中既处理可行个体，也处理不可行个体，而且用 $\mathrm{eval}_u(x)$和 $\mathrm{eval}_f(x)$评价函数来评价它们。

一种方法是用函数 $\mathrm{eval}_f(x)$来设计 $\mathrm{eval}_u(x)$，即 $\mathrm{eval}_u(x)=\mathrm{eval}_f(x)+Q(x)$，其中，$Q(x)$表示一个对不可行个体 x 的惩罚，或者对这样个体的一个修补费用。

另一种方法，可以构造一个全局评价函数 $\mathrm{eval}(p)$，如下：

$$\mathrm{eval}(p)=\begin{cases} q_1\mathrm{eval}_f(p) & \text{if} \quad p\in F \\ q_2\mathrm{eval}_u(p) & \text{if} \quad p\in U \end{cases}$$

用两个权值 q_1 和 q_2 来分别对 $\mathrm{eval}_f(x)$和 $\mathrm{eval}_u(x)$的重要性打分。

上面所描述的方法都允许不可行个体比可行个体“更好”。一般而言，有可能存在一个可行个体 x 和一个不可行个体 y，使 $\mathrm{eval}(y)>\mathrm{eval}(x)$ 成立，这可能会引起算法收敛到不可行解，为了避免这种情况的发生，在实际运算中常常采用动态惩罚，以增加不可行个体在当前搜索状态的选择压力。需要说明的是，选择合适的权值 q_1 和 q_2 通常也有一定的困难。

文献[105]假设任何可行个体都比不可行个体“好”，收到了较好的结果；Powell 和 Skolnick[105]应用此启发式规则于数值优化问题，可行解的评价被映射到（-∞，1）内；不可行解被映射到区间（1,+∞）（对求最小问题）；Michalewicz 和 Xiao[106]实算了路径安排问题，并对可行个体和不可行个体使用了两个分离的评价函数。eval_u 的值通过加入一个常数而增加（减小吸引力），以便使最好的不可行个体比最差的可行个体差。但是这

种策略依然有不尽人意的地方，特别是在图 6-2 所示的情况下，不可行个体“m”实际上要比可行个体“b”更好。因为它“紧邻”最优解，一些策略往往忽略这一点。运用演化算法于一个特定问题时，在可行个体和不可行个体评价函数之间建立一种关系是最具挑战性的问题之一。

3．不可行个体的丢弃

“死亡惩罚”规则在许多演化技术中经常被用到，消除不可行个体使算法得到了某种简化：算法无须设计 eval_u 。当可行搜索空间为凸集，且它占整个搜索空间的比例较大时，从一个群体中消去不可行解的方法能够十分有效。“死亡惩罚”被早期的 ES、EP 和 SA 等方法广泛采用，但当搜索空间复杂或可行空间占整个空间的比例过小时，使用这一策略往往不能达到预期的结果。

4．不可行个体的修补

在演化计算界，修补算法的应用十分广泛：对许多组合优化问题，如货郎担问题、背包问题、集合覆盖问题等。很容易“修补”一个不可行个体，这样的修补也可以只用于评价。

修补不可行个体过程与个体的学习紧密相关，即所谓的 Baldwin 效应[107]。正如一般的局部搜索，特别是对最近可行解的局部搜索，学习和评价彼此相互作用，个体获得改进的适应值。例如，不可行解和与之最近的可行解之间的信息交换，就是一个简单的修补方法。

这些方法的缺点是它们对问题的依赖性。对某个特定的问题，应该设计一种特殊的修补算法。而且没有通用的规则来设计这样的算法，通常可使用贪婪修补、随机修补或者一些其他能指导修补过程的启发式规则。对于一些问题，修补不可行个体的过程也可能和解决原始问题一样复杂，如对非线性约束问题、计划和时间表问题等，这时需要考虑的是：是否应该采用别的方式来处理不可行个体。

Orvosh 和 Davis 报告了[108]一个所谓的 5%规则，它表明在许多组合优化问题中，当用 5%的修补个体替换它们不可行的原始个体时，能获得最好的结果。对于一些特殊问

题，文献[108]中的试验表明 15%替换规则明显好于其他替换率。由此看来，“最优”的替换概率也与问题有关，并可能随演化过程而变化。

5．不可行个体的惩罚

对不可行个体进行惩罚，是演化算法解决约束问题常用的方法，在罚函数法中，$\mathrm{eval}_u(x)=\mathrm{eval}_f(x)+Q(x)$，$Q(x)$表示对不可行个体 x 的罚函数，或者对这样个体的修补费用。通过这种变换，把约束优化问题转化成无约束优化问题。

罚函数法的关键在于惩罚函数 $Q(x)$应该如何设计。惩罚过于严厉会造成新的问题过于复杂，可行的最优点难以被发现。从直觉上，惩罚应该尽可能低，即所谓的最小惩罚规则。然而，这一规则是很难做到的，往往过小的惩罚会起不到“惩罚”效果。设计这样的罚函数是与问题有关的，而且通常是很难的。

惩罚方法的选择依赖于：

① 可行空间和整个搜索空间的大小之间的比率；

② 可行空间的拓扑性质；

③ 目标函数的类型、变量数；

④ 约束的数目及约束的类型。

因此，使用罚函数十分复杂。Homaifar 等[109]提出了一种惩罚策略：对于每一个约束，定义若干违约级别，对每一个违约级别产生一个惩罚系数 R_{ij}， $j=1,2,\mathbf{L},m$ 为约束的个数，$i=1,2,\mathbf{L},l$ 为每个约束违约的级别数。违约的级别越高，惩罚系数 R_{ij} 的值就越大。对于群体中的每一个个体 x 用下式进行评价：

$$\mathrm{eval}(x)=\mathrm{eval}_f(x)+\sum_{j=1}^{m}R_{ij}f_j^2(x) \qquad (6\text{-}1)$$

其中，f_j 是违背第 j 个约束的惩罚函数。这种惩罚方式为静态惩罚。

与上述方法相对，Joines 和 Houck 等[110]提出了一种动态惩罚方式，个体按式（6-2）进行评价：

$$\mathrm{eval}(x)=\mathrm{eval}_f(x)+(C\times t)^a\sum_{j=1}^{m}f_j^b(x) \tag{6-2}$$

其中，C、a、b 为常数，t 为算法迭代的次数。通常，$C=0.5$，$a=b=2$。与式（6-1）相比，这种方法要求的参数较少，并且不定义违约的级别。不可行解的选择压力随着迭代次数 t 的增加不断增大。

Schoenauer 和 Xanthakis[111]提出了如算法 6-1 所示的惩罚方法。

算法 6-1　Schoenauer 和 Xanthakis 提出的惩罚方法

0	Algorithm：Punishment method put forward by Schoenauer and Xanthakis
1	首先，设置约束计数器 j=1；
2	设 $\mathrm{eval}(x)=f_j(x)$ 为评价函数，对群体进行演化操作，直到群体中对此约束可行的个体达到一定比例 f（触发阈值）为止；
3	设置 j=j+1，对新群体进行演化操作，从群体中删除不满足约束 1,2,…,j-1 的个体，直到群体以比例 f 再次满足第 j 个约束；
4	若 j<m，重复 2、3 步骤，否则，用目标函数评价个体，排除不可行个体。

这种方法要求所有约束按线性次序依次被处理，这种次序对算法的执行结果有一定的影响。此方法每次只考虑一个约束，而在算法的最后直接采用目标函数本身评价个体，而不带有任何惩罚因素。

Powell 和 Skolnick[112]提出了另外一种方法，在这个方法中按下式对个体进行评价：

$$\mathrm{eval}(x)=\mathrm{eval}_f(x)+r\sum_{j=1}^{m}f_j(x)+l(t,x) \tag{6-3}$$

其中，r 为常数；$l(t,x)$ 为一个附加函数，可以采用不同的启发规则，如“任何可行个体都好于不可行个体”，可通过下式完成：

$$l(t,x)=\begin{cases}0 & \text{if} \quad x\in F\\ \max\left\{0,\max\limits_{x\in F}\{f(x)\}-\min\limits_{x\notin F}\left\{f(x)+r\sum\limits_{j=1}^{m}f_j(x)\right\}\right\} & \text{if} \quad x\notin F\end{cases}$$

上述策略对于求解约束优化问题，都有广泛的借鉴意义。

6．用特殊的表达和遗传算子维持可行群体

还有一种处理不可行性问题的方法是使用特殊的表达和遗传算子，以维持群体中个体的可行性。例如，假设问题的可行搜索空间是凸的，$\boldsymbol{x}$ 和 $\boldsymbol{y}$ 是该搜素空间上的两个可行解，则其算术杂交：$a\boldsymbol{x}+(1-a)\boldsymbol{y}$，其中，$0 \leqslant a \leqslant 1$，在凸搜索空间总产生一个可行解。因此，没有必要定义函数 $\mathrm{eval}_u(x)$；这样的系统对目标函数及搜索空间的要求较高，但通常比其他基于惩罚方法的演化技术更可靠，因此，一些研究者用与问题有关的表达和特殊算子在许多领域中构造出了非常成功的演化算法——文化算法。

6.2 文化算法简介

6.2.1 文化算法框架

Robert G. Reynolds 和 Xidong Jin 描述了一种文化算法[113]，这种方法粗糙地模拟了人类社会的演化过程。在人类社会中，文化被看作信息的载体，可以被社会所有成员全面接受，并用于指导每个社会成员的各种行为。在文化算法中，包含有两个演化空间：一个是由在进化过程中获取的经验和知识组成的信仰空间（Belief Space）；另一个是由具体个体组成的群体空间（Population Space）。这两个空间通过特定的协议进行信息交流。文献[113]描述文化算法的具体构架如图 6-5 所示。

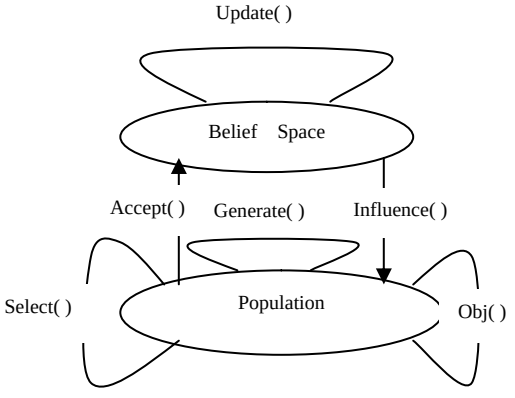


图 6-5 文化算法框架

其中，Accept( )函数用于收集从群体中被选择个体的经验；Influence( )函数可以利用解决问题的知识指导群体空间的演化。信念空间也可以通过 Update( )函数改进。而 Obj( )函数和 Select( )分别是群体的操作函数（如交叉、变异等）和群体选择函数。算法 6-2 描述了文化算法的执行步骤。

算法 6-2　文化算法

0	Algorithm: Culture Algorithms, CA
1	Let t = 0, Initialize Population POP(t) and Belief Space BLF(t);
2	Do
2.1	Evaluate Population POP(t);
2.2	Adjust(BLF(t), Accept(POP(t)));
2.3	Adjust(BLF(t));
2.4	Variation(POP(t) from POP(t-1));
3	Until termination criterion is met

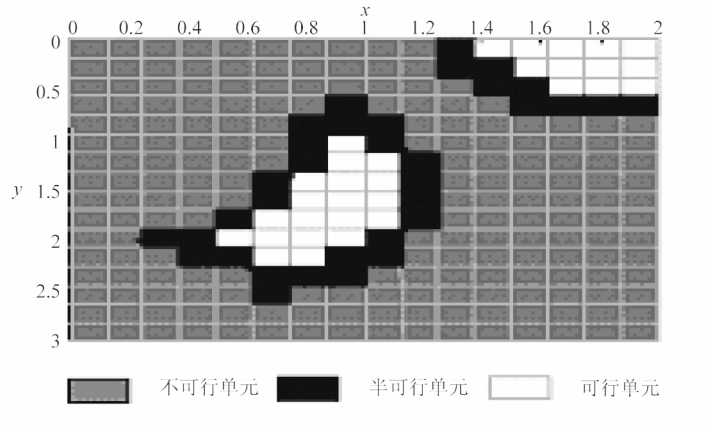
6.2.2　信仰空间的约束表达和信仰空间的更新

1．信仰空间的约束表达

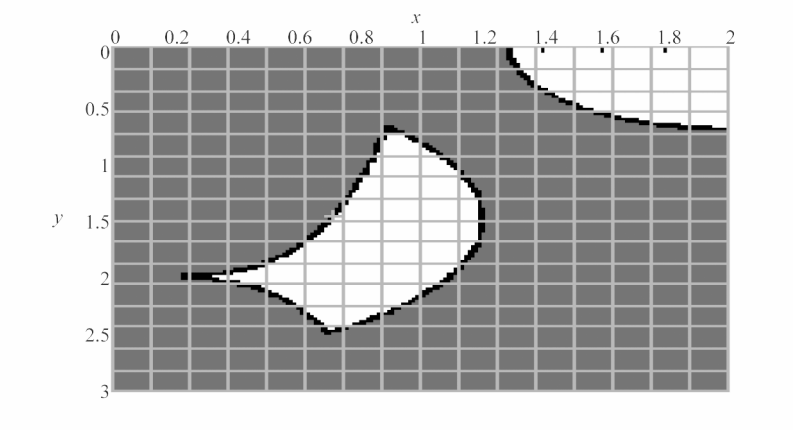
在解决约束优化问题时，一个关键问题是如何将问题的约束表示成信仰空间中的知识并保存起来。

事实上，问题的约束总是将搜索空间分割成较小的区间，不同区间有不同的特性，一些区域是可行的，满足所有的约束条件，相应的另一些区域是不可行的。所以，可以将搜索空间分成更小的区间，这些区间称为单元（Cell），每个单元也有不同的特性：一些单元完全在可行（不可行）区域，是可行（不可行）的，另外一些由于既在可行区域又在不可行区域，是半可行的。文献[113]称这样的单元为信仰单元（Belief Cells），这些信仰单元构成了文化算法的信仰空间，用来表示、存储问题的约束知识，并提供信仰单元的约束知识，指导群体空间进化。

如图 6-6 所示为二维的搜索空间被分割成信仰单元的示例。在图 6-6（a）中，非线性约束将域空间分成不同的形状。在图 6-6（b）中，可行/不可行区域知识可以根据信仰单元显式地辨别出来。



（a）



（b）

图 6-6　域空间和信仰单元的实例

假设问题搜索空间的维数是 n，信仰空间的结构定义为〈$N[n]$,$C[m]$〉，其中 $N[j]$是关于维度 j 的某一区间信息，$N[j]$可表示为〈I_j , L_j , U_j〉（$j=1, \cdots, n$），I_j 为一个闭的区间：$I_j=[l_j, u_j]=\{x \mid l_j \leqslant x \leqslant u_j, x \in R\}$，$L_j$ 和 U_j 分别表示 l_j 和 u_j 评价函数值；C 是一个信仰单元集合，m 是信仰单元数，在进化过程中保持静态或随系统变化。$C[i]$表示为〈Class_i , Cntl_i , Cnt2_i , W_i , Pos_i, Csize_i〉。Class_i 表示第 i 个信仰单元的性质：可行、不可行、半可行或未知。Cntl 和 Cnt2 是信仰单元中的两个计数器，前者计算信仰单元的可行个体侯选者的数量，后者计算信仰单元的不可行个体侯选者的数量，两个计数器还可以提供分类可靠性的信息，它们都初始化为 0。W_i 是第 i 个信仰单元的权重，可信度高的信仰单元具有较高

的权重。Pos_i是一个表示信仰单元 i 的最左角的位置。$Csize_i$是一个表示信仰单元的大小。

实际上，信仰空间模拟了 n 维空间中“大小随时变化的超立方体”，这个立方体由更小的信仰单元构成，通常情况下，可以有很多这样的活动的立方体。一般，在这个立方体中可行和半可行的信仰单元比其他区域更具有吸引力。

2．更新信仰空间

accept( )函数在群体空间内选择可以直接影响当前信仰空间的个体。在文化算法中标准化的知识和约束知识被不同的个体影响。本书中，accept( )函数按照评价函数值，选择前 20%的个体来重新修正信仰空间。

文化算法通过 update( )函数对信仰空间〈$N[n],C[m]$〉进行更新。

假设第 i 个个体影响 $N[j]$ 的下限。第 k 个个体影响 $N[j]$ 的上限，通过下式对 $N[j]$ 进行修正：

$$l_j^{t+1}=\begin{cases} x_{i,j}^t & \text{if } x_{i,j}^t \leqslant l_j^t \text{ or } \mathrm{obj}(x_i^t) < L_j^t \\ l_j^t & \text{otherwise} \end{cases}$$

$$L_j^{t+1}=\begin{cases} \mathrm{obj}(x_i) & \text{if } x_{i,j} \leqslant l_j^t \text{ or } \mathrm{obj}(x_i) < L_j^t \\ L_j^t & \text{otherwise} \end{cases}$$

$$u_j^{t+1}=\begin{cases} x_{k,j} & \text{if } x_{k,j} \geqslant u_j^t \text{ or } \mathrm{obj}(x_k) < U_j^t \\ u_j^t & \text{otherwise} \end{cases}$$

$$U_j^{t+1}=\begin{cases} \mathrm{obj}(x_x) & \text{if } x_{k,j} \geqslant u_j^t \text{ or } \mathrm{obj}(x_k) < U_j^t \\ U_j^t & \text{otherwise} \end{cases}$$

其中，l_j^t、u_j^t表示 $N[j]$的第 t 代的下限和上限，L_j^t和 U_j^t表示它们的评价函数值。群体中的每个个体提供它们所在的信仰单元的约束信息。对于一个信仰单元 i，$Cnt1_i$记录着有效个体数，$Cnt2_i$记录着无效的个体数。

$$\mathrm{Class}_i=\begin{cases} \text{unknown} & \text{if } \mathrm{Cnt1}_i=0 \text{ and } \mathrm{Cnt2}_i=0 \\ \text{fesible} & \text{if } \mathrm{Cnt1}_i>0 \text{ and } \mathrm{Cnt2}_i=0 \\ \text{unfesible} & \text{if } \mathrm{Cnt1}_i=0 \text{ and } \mathrm{Cnt2}_i>0 \\ \text{semi-feasible} & \text{otherwise} \end{cases}$$

W_i 根据储存在 Class_i 中的值直接更新。Csize_i 表示信仰单元的大小，保持 N 中信仰单元的数量不变，那么信仰单元的大小可以根据 N 尺寸改变。本书即采取这种方法。信仰空间通过 influence( )函数对群体空间施加影响，将个体从不确定单元移到确定单元中。基本过程如下。

① 如果群体中的个体不在目前〈$N[n],C[m]$〉所确定的超立方体之内，将通过下式把其移动到这个立方体中。，

$$x_{nc+i,j}=\begin{cases}x_{nc,j}+\left|(u_j-l_j)N_{nc,j}(0,1)\right| & \text{if } x_{nc,j}<l_j\\ x_{nc,j}-\left|(u_j-l_j)N_{nc,j}(0,1)\right| & \text{if } x_{nc,j}>u_j\end{cases} \quad \forall i\in\{1,2,...,nc\},\ \forall j\in\{1,2,...,np\}$$

② 如果个体在这个超立方体内，通过下式对个体的位置进行修正：

$$x_{nc+i,j}=\begin{cases}\text{moveTo}(\text{choose}(\text{Cell}[m])) & \text{if } x_{nc,j}\in\{\text{unfeasible cells}\}\\ x_{nc,j}+\left|(u_j-l_j)\times N_{nc,j}(0,1)\right|/m_j & \text{otherwise}\end{cases}$$

$$\forall i\in\{1,2,...,nc\},\ \forall j\in\{1,2,...,np\}$$

其中，m_j 是信仰单元的个数，choose(Cell[m])用来选择更有希望的目的单元。这里，这里用轮盘赌实现这种选择。定义权重函数 Cell[k]如下：

$$W_k=\begin{cases}w_1, & \text{if Cell}[k]\in\{\text{unknown cells}\}\\ w_2, & \text{if Cell}[k]\in\{\text{feasible cells}\}\\ w_3, & \text{if Cell}[k]\in\{\text{semi}-\text{feasible cells}\}\\ w_4, & \text{if Cell}[k]\in\{\text{unfeasible cells}\}\end{cases}$$

其中，$w_1=w_2=2$，$w_3=3$，$w_4=1$。

moveTo( )是迁移函数，将个体移到被选单元，假设个体 i 应移到 Cell[k]，moveTo(Cell[k])定义如下：

$$x_{nc+i,j}=\text{Pos}_{k,j}+\text{Uniform}_{nc,j}(0,1)\text{Csize}_{k,j}$$

其中，$\text{Pos}_{k,j}$是Cell[k]的位置，$\text{Csize}_{k,j}$是Cell[k]的尺寸，$\text{Uniform}_{nc,j}(0,1)$通过正态分布产生[0,1]中的一个数。

文化算法通过上述办法完成约束的表达，并通过几个函数实现群体空间和信仰空间的通信，算法通过群体空间的个体中蕴涵的信息来不断地修正信仰空间，同时信仰空间指导群体中的个体在最有希望的区域进行搜索。

6.2.3　群体空间的演化

群体空间与信仰空间的演化关系，类似于自然进化和文化发展之间的相互作用。群体空间的变化通过相应的传递函数影响文化的发展，两个空间的变化由此形成了一个有机的整体。在群体空间上，可以采用之前介绍的任何一种演化计算方法，这里就不再过多阐述，在文献[115]中采用的是标准的进化规划（CEP），本章试验中也是如此。

6.3　算法测试

选择附录 B 中提供的约束优化问题，对文化算法进行测试。在数值试验中，群体规模为 100，对于每个问题在相同的条件下独立运行 20 次，由于 *T*1 和 *T*2 的维数较小，运行 1000 次，而对于其他问题，运行 2500 次，平均结果如表 6-1 所示。

从表 6-1 中可以看出，对于 *T*1 和 *T*2，问题维数较低，CA 方法都找到了全局最优值；对于 *T*3、*T*4，CA 方法所获得的结果依然十分接近全局最优值；*T*5 为二次函数，并有 9 个线性约束，其中约束 4、5、6 在全局最优点附近并没有起到作用，算法多次找到全局最优值，平均最优值的误差也较小；*T*6 本身非常简单，但其中含有 3 个线性约束和 3 个非线性约束，这 6 个约束在全局最优处都起作用，20 次运行中，算法没有找到问题的全局最优值，平均最优值与实际最优值有一定的误差；与其他问题不同，*T*7 中包括 3 个非线性等式约束，CA 得到的结果没有什么实际意义，这种情况是否与等式约束有关，目前还不能下结论，需要进一步研究和实验；*T*8 包括 3 个线性约束和 5 个非线性约束，其中，前 6 个约束在全局最优处起作用，得到的结果令人比较满意。

表 6-1　CA 方法与文献实验结果

Func.	f^*	Culture Algorithms	
		Best	Mean
*T*1	1.3934651	1.393464	1.393464
*T*2	-6961.81381	-6961.813	-6961.820
*T*3	680.630057	680.630	680.645
*T*4	-30665.538	-30665.818	-30647.538
*T*5	-15	-15	-15.002
*T*6	7049.330923	7017.784450	6909.214301
*T*7	0.0539498478	—	—
*T*8	24.3062091	24.306129	27.120778

应用文化算法处理约束优化问题具有一定的潜力，信仰空间可以表示、存储和集成约束知识和其他基于域的知识，为群体提供指导。目前，在国内关于 CA 的研究较少。本节中的试验结果证实文化算法在一些约束优化问题上的有效性。

6.4　小结

针对约束优化问题，目前还没有一种现成的方法来确定问题的全局最优解。本书概括了几种用演化方法求解约束优化问题的策略，着重介绍了不可行个体的处理方式。其中包括：

① 维持群体中个体的可行性；

② 使用特殊的算子或编码方式；

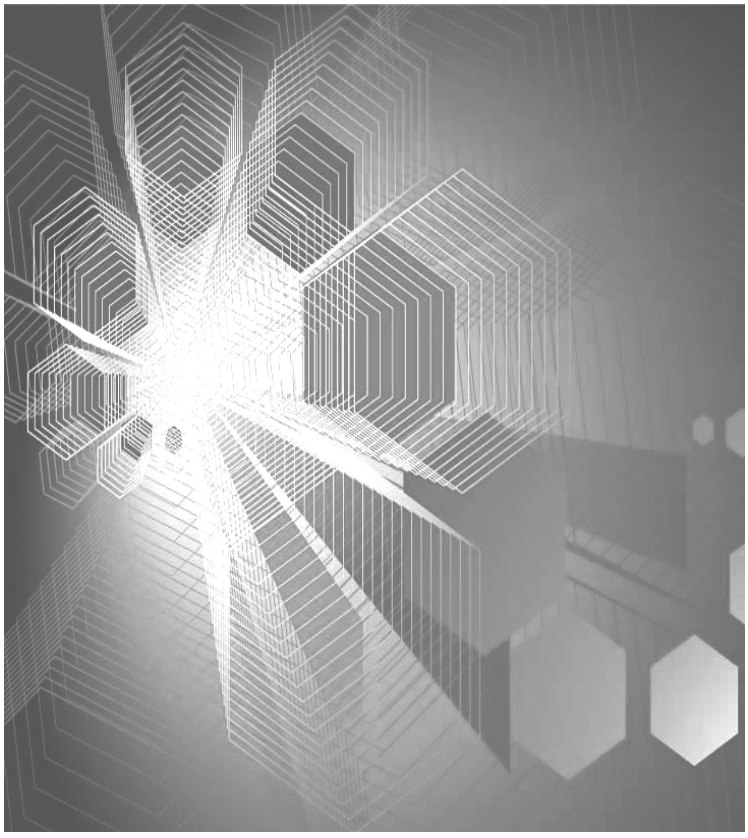
③ 可行个体优先原则（任何一个可行个体比所有不可行个体都要“好”）；

④ 以一个特定的线性次序来考查约束、修补或惩罚不可行个体等。

本书引入了文化算法，国内目前还没有关于这种算法的研究，这种方法粗糙地模拟了人类社会的进化过程。在人类社会中，文化被看作信息的载体，可以被社会所有成员全面地接受，并用于指导每个社会成员的各种行为。在文化算法中，包含两个演化空间：一个是由在进化过程中获取的经验和知识组成的信仰空间（Belief Space）；另一个是由

具体个体组成的群体空间（Populatien Space）。在解决约束优化问题时，将问题的约束表示成信仰空间中的知识并保存起来，以指导另一个由具体个体组成的群体空间（Population Space）上的演化。这两个空间通过特定的协议进行信息的交流。在信仰空间的启发下，任何一种求解无约束优化问题的算法都可以用于求解约束优化问题。因此，应用文化算法处理约束数值优化问题还具有很大的潜力。

第 7 章　蚁群优化



7.1　蚁群优化算法

用“蚂蚁觅食”来描述蚁群在“觅食”过程中的生物行为。假设一个蚂蚁群体中有 m 只蚂蚁，单个蚂蚁能将 A 处的“食物”搬运到巢穴 B 处，每只蚂蚁在运动中会在所经过的路径上留下一种挥发性分泌物（以下称为“信息素”），并且通过信息素来实现个体之间的“通信”，发现并且运食物回来的蚂蚁 k_1 遇见迎面而来的 k_2，k_1 “告诉” k_2 自己从 A 处搬运“食物”回巢穴 B 处，即将信息传给对方。在蚂蚁觅食的过程中，每只蚂蚁的行动是随机、并行的。开始时各个蚂蚁的搬运是无序的，但经过一段时间以后，由于个体之间的“通信”，便逐渐形成了有序的搬运、堆放这样一种群体活动。在“蚂蚁觅食”的过程中，路径的选择是由信息素的浓度决定的，并倾向于朝信息素浓度高的方向移动，而信息素的浓度关键在于单位时间通过的蚂蚁数量。假设 A、B 之间有两条路径 a、b，路径 a 的距离比 b 短，蚁群由 B 至 A 再返回 B。在通常情况下，经过一段时间后，会有较多的蚂蚁选择 a。但是假如 b 比 a 的“路况”要好，一段时间后，选择 b 比选择 a 来回的时间要短（比如 a 路径发生了堵塞），这样蚂蚁留下的信息素会随着时间的流逝而“挥发”，b 的信息素浓度反而比 a 大。一段时间后，选择 b 路径的蚂蚁反而会比 a 路径多。

可以看出，在“蚂蚁觅食”过程中，蚂蚁会在经过的路径上留下信息素，信息素随时间的推移会逐渐挥发消失。蚂蚁在运动中能感知这种物质的存在及浓度，并倾向于朝信息素浓度高的方向移动。即选择该路径的概率与当时这条路径上该物质的浓度成正比。信息素浓度越高的路径，选择它的蚂蚁就越多，则在该路径上留下的信息素的浓度就更大；而浓度大的信息素又吸引更多的蚂蚁，从而形成一种正反馈。正是通过这种正反馈，蚂蚁最终可以发现最佳路径，并且大部分蚂蚁都会走此路径。

用于寻找最优路径的蚁群算法来源于蚂蚁觅食的行为，蚁群算法为传统复杂问题的求解提供了一种新的方法。例如，TSP（旅行商）问题，它可以表述为在 n 个城市中寻找一条访问每一个城市且每个城市只能被访问一次的最短长度的闭环路径。现在以求解 n 个城市的 TSP 问题为例来说明基本蚁群（AS）算法。对于其他的问题，只需要对此算法稍作修改即可应用。

设蚂蚁的数量为 m，$D_{i,j}$ 表示城市 i 和城市 j 之间的距离，用 $t_{i,j}(t)$ 表示 t 时刻城市 i 和 j 之间路径上残留的信息素浓度。初始时刻，各条路径上的信息素浓度是相等的，设为 $t_{i,j}(0)=C$（C 为常数）。在蚂蚁的运动过程中，其转移的方向是由信息素浓度来决定的。在蚁群算法中，需要一个蚂蚁转移的概率 $p_{i,j}^{k}(t)$，它表示在 t 时刻，蚂蚁 k 由城市 i 转移到城市 j 的概率。

$$p_{i,j}^{k}(t)=\begin{cases}\dfrac{[t_{i,j}(t)]^{a}[h_{i,j}]^{b}}{\sum\limits_{j\notin \mathrm{tabu}_k}[t_{i,j}(t)]^{a}[h_{i,j}]^{b}} & j\notin \mathrm{tabu}_k \\ 0 & j\in \mathrm{tabu}_k\end{cases} \tag{7-1}$$

其中，$h_{i,j}$ 为经验知识（或称为“能见度”），一般取 $h_{i,j}=1/D_{i,j}$。a 、b 是控制信息素浓度与可见度的重要参数，且是非负的，它们决定了 $t_{i,j}(t)$ 和 $h_{i,j}$ 对蚂蚁转移概率的影响。而 tabu_k（$k=1,2,\mathbf{L},m$）用以记录蚂蚁 k 当前所走过的城市。由于在这个问题中，蚂蚁走过一个城市以后，下次就不能再到这个城市，集合 tabu_k 需要随着蚂蚁到达每一个城市后作动态增加，把 tabu_k 称为第 k 只蚂蚁的禁忌表。这个表中保存的城市蚂蚁在移动中将不能再去。随着时间推移，以前留下的信息素会逐渐挥发。因此，当完成一次遍历后，各路径上的信息素浓度要根据式（7-2）进行调整：

$$t_{i,j}(t+1)=r\,t_{i,j}(t)+\Delta t_{i,j}(t,t+1) \tag{7-2}$$

其中 $r\in(0,1)$，$(1-r)$ 为信息素浓度的衰减系数，在本次遍历 $(t,t+1)$ 时间段中信息素浓度的增量 $\Delta t_{i,j}(t,t+1)$ 表示为

$$\Delta t_{i,j}(t,t+1)=\sum_{k=1}^{m}\Delta t_{i,j}^{k}(t,t+1) \tag{7-3}$$

$\Delta t_{i,j}$ 表示蚂蚁 k 在本次遍历 $(t,t+1)$ 时间段中留在路径〈i, j〉上的信息素浓度，它一般用下式表示：

$$\Delta t_{i,j}^{k}(t,t+1)=\begin{cases}Q/L_k & \text{，蚂蚁}k\text{游历过〈}i,\ j\text{〉} \\ 0 & \text{，其他}\end{cases} \tag{7-4}$$

其中，Q 为常数，L_k 表示蚂蚁 k 在本次遍历中所走过的路径的长度。

基本蚁群算法如算法 7-1 所示。

算法 7-1　蚁群优化算法

0	Ant Colony Optimization，ACO
1	在 0 时刻进行初始化过程，蚂蚁放置在不同的城市，每一条边都有一个初始信息素浓度值 $t_{i,j}(0)$。每只蚂蚁的禁忌表的第一个元素置为它的开始位置。
2	每一只蚂蚁从位置 i 移动到位置 j，依据信息素浓度与能见度两个重要参数（包括参数 a 、b ）来选择移动新位置（见式（7-1））。当所有蚂蚁都完成了一次遍历，它们的禁忌表被装满；这时，计算每一只蚂蚁 k 的 L_k，同时根据式(7-2)和式(7-3)来计算 $\Delta t_{i,j}(t,t+1)$、$\Delta t_{i,j}^{k}(t,t+1)$，并更新 $t_{i,j}(t+1)$。此时由蚂蚁找到的最短路径（ $\min L_k$， $k=1,2,\mathbf{L},m$ ）将被保存，所有禁忌表将置空。当前每只蚂蚁所在位置将置入相对应的禁忌表。
3	重复执行步骤 2，直至满足算法结束条件，即这一过程重复直到周游计数器达到最大周游数 $\mathrm{NC}_{\max}$（用户定义），或者所有蚂蚁都走同一路线。

蚁群算法是一种本质并行的算法。蚂蚁搜索的过程彼此独立，只通过信息素进行间接通信。众所周知，并行计算可以显著减少计算时间，所以，蚁群算法在计算量较大的情况下是一种可以选择的好方法。

蚁群算法是一种正反馈算法。在整个过程中任何一段路径上的外激素水平较高，将吸引更多的蚂蚁选择这条路径运动，而这又将会使得本条路径的外激素水平增加。可以这样认为，由于正反馈的存在，将会使得整个搜索过程加速收敛。

蚁群算法的健壮性较好。相对于其他算法，蚁群算法对初始路线的要求不高，也即初始条件的设置对将来结果的取得不会有太大的影响，蚁群算法的搜索结果不依赖于初始线路的选择。

因此，蚁群算法可以在不需要人工干预的情况完成从初始化到得到搜索结果的整个计算过程。

由于蚁群算法具有并行性、正反馈、健壮性等特点，并且搜索过程不需要人工干预，在解决小规模（n<30）的旅行商问题时效果显著。但对于规模较大的问题，其性能却迅速恶化。主要原因是算法的初始阶段，各条道路上的信息素水平基本相等，蚂蚁的搜索呈现出较大的盲目性。只有经过较长时间后，信息素水平才呈现出明显的指导作用。另外，由于蚁群算法是一种正反馈算法，在算法速度收敛较快的同时，也容易陷入局部优

化，导致其在处理大规模旅行商问题时性能下降明显。蚁群算法的另一不足是，许多参数的设置凭借经验，没有充足的依据，蚂蚁数量的设定往往依赖于实验结果进行调整，应用蚁群算法求解旅行商问题时，需要进行实验，根据结果调整参数，然后再进行搜索，但这不利于算法的广泛应用。

7.2 蚁群聚类

在数据挖掘中，聚类是一个活跃的研究领域，目前存在多种聚类方法，这些方法不仅算法原理不同，而且许多基本特性也不相同。1991 年，Deneubour 等介绍了基于蚂蚁的聚类和分类方法，当时主要用于机器人作业调度，Lumer 等修改了这个算法并将之应用于对数字数据分析中。后来这个算法应用于数据挖掘、图像分割和文本挖掘中。2002 年，Labroche 等提出基于蚂蚁化学识别系统的聚类方法。总体说来，基于蚁群算法的聚类方法从原理上可以分为如下 4 种：

① 运用蚂蚁觅食的原理，利用信息素来实现聚类；

② 利用蚂蚁自我聚集行为聚类；

③ 基于蚂蚁堆的形成原理实现数据聚类；

④ 运用蚁巢分类模型，利用蚂蚁化学识别系统进行聚类。

产生蚁群聚类算法的灵感，源于蚂蚁如何堆积它们的尸体和如何进行幼体分类。Chretien 等人用蚂蚁做了大量实验，发现工蚁能在几小时内将分散在蚁穴各处的大小不同的蚂蚁尸体聚成几类。小的蚁堆通过吸引蚂蚁积攒更多的尸体来逐渐变大，这种正反馈会导致蚁堆越积越大，以达到聚类数据的目的。另外，观察还发现，蚂蚁能将大小不同的蚁卵安排在不同位置。Deneubourg 等人通过对这些实验的观察和研究，提出了一种解释蚁群聚类现象的基本模型（Basie Model，BM），并模拟实现了蚁群的聚类过程。此后，Lumer 和 Faieta 将基本模型推广并应用于数据分析，提出了 LF 算法。LF 算法的主要思想是：先将所有多维属性空间中的数据对象随机地投影到二维网格平面上，然后每只蚂蚁在二维平面上随机选择一个数据对象，随即蚂蚁计算该数据对象与邻域半径内

其他数据对象之间在属性空间中的相似性。如果不相似，蚂蚁将数据对象拾起并随机移往别处，再计算与邻域半径内对象的相似性，直到移往与周围对象相似的地方被放下，并随机选择下一个数据对象，如果相似，蚂蚁将不会拾起该数据对象，而随机选择下一个数据对象。这样，数据对象通过被蚂蚁的不断拾起、移动与放下而被聚类。或者说，相似的对象被蚂蚁搬运集中到一起。简单来说，其思想就是不相似的搬走，相似的放下。

设蚂蚁在时刻 t 于位置 r 发现对象 o_i，则 o_i 在 r 处与其邻域内对象 o_j 的平均相似度定义为

$$f(o_i)=\max\left\{0,\frac{1}{s^2}\sum_{o_j\in \mathrm{Neigh}_{s\times s}(r)}\left[1-\frac{d(o_i,o_j)}{a}\right]\right\} \tag{7-5}$$

其中，s 为邻域的边长，$\mathrm{Neigh}_{s\times s}(r)$ 表示位置 r 周围面积为 $s\times s$ 的正方形邻域。$d(o_i,o_j)$ 表示对象 o_i 与对象 o_j 在属性空间中的距离，常用欧氏（Euclidean）距离或余弦距离公式（也可采用其他距离公式）计算。a 为相似性参数，它给出了“相似”的距离范围。需要注意的是，a 在 LF 算法中是一个关键参数，它直接影响聚类结果和算法的收敛速度。假如 a 太大，会把不相似的对象聚为同类，算法快速收敛。假如 a 太小，导致本应聚类的对象不能聚在一起，并且算法收敛缓慢。

蚂蚁拾起或放下一个数据对象是由概率转换函数（Probability Conversion Function）决定的。概率转换函数将相似度转换为蚂蚁拾起或放下数据对象的概率，因此，它是一个以相似度为变量的函数，值域为 $[0,1]$ 的函数。概率转换函数制定的主要原则是相似度越大，拾起概率越小，放下概率越大；相似度越小，拾起概率越大，放下概率越小。

这样，在每次循环中，蚂蚁拾起或放下一个对象都遵循以下原则：如果一只蚂蚁没有负载，那么它会随机地选择一个未被其他蚂蚁选择的对象，计算拾起概率。

$$P_p(o_i)=\left(\frac{k_1}{k_1+f(o_i)}\right)^2 \tag{7-6}$$

如果 P_p 大于一个随机概率 P_r，则蚂蚁拾起该对象；否则，蚂蚁随机选择其他对象。

如果一个蚂蚁有负载，计算放下概率：

$$P_d(o_i)=\begin{cases}2f(o_i) & f(o_i)<k_2\\ 1 & f(o_i)\geqslant k_2\end{cases} \tag{7-7}$$

如果 P_d 大于一个随机概率 P_r，则蚂蚁放下该对象；否则，蚂蚁负载该对象随机移往别处。式（7-6）、式（7-7）中，k_1、k_2 为阈值常量。可以看出，蚂蚁间通过使用相似度进行间接通信。蚂蚁在邻域内放下一个相似的对象，会提高邻域内对象的相似度，而从邻域内搬走一个不相似的对象，同样会提高邻域内对象的相似度。

LF 蚁群聚类算法如算法 7-2 所示。

算法 7-2　LF 蚁群聚类算法

0	Algorithm: Lumer and Faieta Ant　Colony Clustering Algorithm
1	初始化迭代次数 number_cycle，蚂蚁个数 number_ant，网边长 L，邻域边长 s，相似性参数 α，以及常量 k_1、k_2 等，将所有数据对象随机投影到二维网格，每只蚂蚁随机选择一个数据对象；
2	For i = 1 to number_cycle do
2.1	For 每只蚂蚁 do
2.1.1	根据式（7-5）计算相似度 $f(o_i)$；
2.1.2	If 蚂蚁未负载 then
2.1.2.1	根据式（7-6）计算拾起概率 P_p；
2.1.2.2	If $P_p > P_r$ then
2.1.2.2.1	蚂蚁负载该对象随机移往其他位置；
2.1.2.3	Else
2.1.2.3.1	蚂蚁随机选择另一对象； Endif
2.1.3	Else　//如果蚂蚁负载
2.1.3.1	根据式（7-7）计算放下概率 P_d；
2.1.3.2	If $P_d > P_r$ then
2.1.3.2.1	蚂蚁放下该对象，随机选择另一对象；
2.1.3.3	Else
2.1.3.3.1	蚂蚁负载该对象随机移往别处； Endif
	Endif
2.2	End for
3	End for
4	For 每一数据对象
4.1	邻域内数据对象递归的聚类；
5	End for

LF 蚁群聚类算法在处理蚂蚁拾起或放下数据对象时，采用概率转换函数的方法，即当拾起概率转换函数的值大于某一随机概率时，数据对象被拾起；当放下概率转换函数的值大于某一随机概率时，数据对象被放下。然而，概率转换函数又是一个以相似度作为自变量的函数，这样会存在如下弊端。

① 当蚂蚁未负载时，选择了一个数据对象，如果该数据对象与邻域内数据对象相似，根据相似度的公式计算后，相似度较大，再代入概率转换函数的拾起概率公式，计算后拾起概率应该较小，但当该拾起概率恰好大于一个随机生成的概率时，会导致相似的对象被拾起并搬走。

② 当蚂蚁负载一个数据对象到某处，与该处邻域内的数据对象相似，按照放下概率公式计算后，放下概率应该较大，但当放下概率恰好小于一个随机生成的概率时，同样会导致相似的对象本应放下却被搬走。

③ 当蚂蚁未负载，选择了一个与邻域内对象不相似的对象，本应被拾起，即拾起概率较大，但当拾起概率恰好小于随机生成的概率，这样导致不相似的对象未被拾起。

④ 当蚂蚁负载一个对象到与邻域对象不相似的地方，本应继续负载移动，即放下概率较小，但当放下概率恰好大于随机生成的概率，这样又导致不相似的对象被放下。

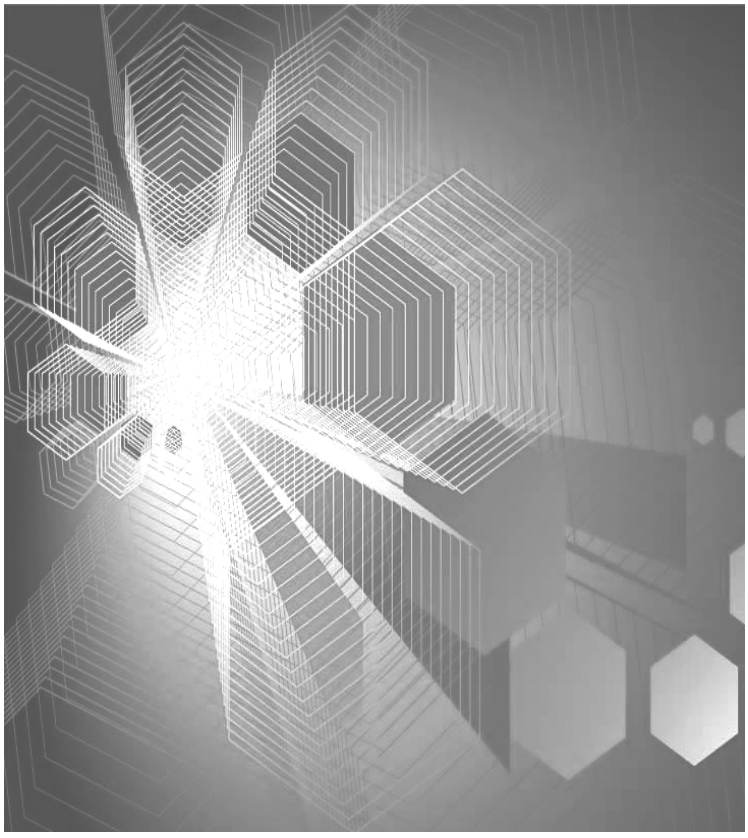
对于以上问题，一些研究者给出了参数的自适应设置方法，能够比较好地解决蚂蚁聚类算法中的收敛速度和聚类质量之间的矛盾。由于自适应地修改参数，算法对参数取值的限制较少，使得计算成本减小，速度加快，对于解决高维的、复杂的数据聚类问题也十分有效。

7.3　小结

蚁群优化算法（Ant Colony Optimization，ACO）是受蚂蚁集体行为启发而提出的一种基于种群模拟的进化算法。研究者经过大量的观察研究发现，蚂蚁个体之间可以通过分泌信息素（Pheromone）来传递信息。蚂蚁在行动的过程中，会在经过的路径上留下信息素，后面的蚂蚁通过感知这种物质的浓度来选择自己的路径。这样，由大量蚂蚁

组成的蚁群集体行为就表现出了一种信息正反馈的现象，信息素随时间挥发，在较短的路径上浓度较大，因而，蚁群总是可以找到更短的路径取食。用人工蚂蚁来模仿自然蚂蚁，在走过的路径上留下信息素，为解决各种寻优问题提供了一种新的方法。该算法已经被成功地应用在很多复杂的组合优化问题上。意大利学者 Macro Dorigo 首先将该方法用于求解著名的旅行商问题（TSP），随后诸多学者陆续使用了该算法解二次分配问题、皇后问题等。所解决的问题以各种 TSP 问题为主，另外有函数优化问题、背包问题等，而且从离散空间扩展到了连续空间。在解决这些问题的性能方面，较之传统的优化算法，蚁群优化算法表现出了良好的性能。

第 8 章　应用举例



8.1 属性约简

8.1.1 信息系统与属性约简

一个信息系统可由四元组 $S=(U,A,V,f)$ 表示，其中，U 为所有对象构成的非空有限集合，称为论域；A 为属性集；$V=\bigcup_{a\in A}V_a$，V 是属性值域，V_a 是属性 a 的值域；$f:U\times A\to V$ 是一个信息函数，它为每个对象的每个属性赋予一个信息值，即 $\forall a\in A, x\in U, f(x,a)\in V_a$。信息系统也称为知识表达系统，通常也用 $S=(U,A)$ 来代替 $S=(U,A,V,f)$。

一个知识系统可以用决策表的形式表示，决策表的行对应要研究的对象，列对应对象的属性，对象的信息是通过指定对象的各属性值来表达的。决策表是一类特殊而重要的知识表达系统，多数决策问题都可以用决策表的形式来表达。决策表可以根据知识表达系统定义如下。

设 $S=(U,A,V,f)$ 是一个知识表达系统，$A=C\cup D$，且 $C\cap D\neq\varnothing$，C 称为条件属性集，D 称为决策属性集，V 是属性的值域，f 是对象属性到值域的映射。具有条件属性和决策属性的知识表达系统称为决策表。

在决策表中，不同的属性可能具有不同的重要性。为了找出某些属性（或属性集）的重要性，一般的方法是从表中去掉一些属性，再来考查没有该属性后分类会怎样变化。若去掉该属性后相应的分类变化较大，则说明该属性的重要性较高；反之，说明该属性的重要性较低。所谓属性约简，就是在保持信息系统的分类能力不变的前提下，删除其中的冗余属性。

8.1.2 常用的属性约简方法

到目前为止，研究者提出了许多属性约简算法。下面介绍几种常用的属性约简方法。

1．基于特征矩阵的属性约简

定义 8-1　决策信息系统 $S=(U,C\cup D)$ 关于条件属性子集 $R\subseteq C$ 的特征矩阵为 $\boldsymbol{M}_R=(m_{ij})_{m\times n}$，其中

$$m_{ij}=\begin{cases} c & ,c(u_i)\neq c(u_j)\ \text{and}\ D(u_i)\neq D(u_j),\ u_i,u_j\in U \\ \varnothing & ,\text{其他} \end{cases} \tag{8-1}$$

定义 8-2　设矩阵 $\boldsymbol{X}=(x_{ij})_{m\times n}$，$\boldsymbol{Y}=(y_{ij})_{m\times n}$，则 $\boldsymbol{X}\oplus\boldsymbol{Y}=(y_{ij}\cup y_{ij})_{m\times n}$。

定义 8-3　$a\in C$，$V\subseteq C$，a 相对于 $\boldsymbol{M}_V$ 的互补元素为 $T_{ij}=\{m_{ij}(a):m_{ij}(v)=\varnothing$ 且 $m_{ij}(a)=a\}$，$m_{ij}(v)$ 和 $m_{ij}(a)$ 分别为 $\boldsymbol{M}_V$ 和 $\boldsymbol{M}_a$ 的元素。互补元素越多，说明 a 与 V 互补性越强。

对决策信息系统 $S=(U,C\cup D)$，存在如下性质：

① 设 $C=\{c_1,c_2,\cdots,c_n\}$，则 $\boldsymbol{M}_C=\boldsymbol{M}_{C1}\oplus\boldsymbol{M}_{C2}\oplus\cdots\oplus\boldsymbol{M}_{Cn}$；

② 设 $R\subseteq C$，若 $\boldsymbol{M}_R$ 中不存在空集元素 $\varnothing$，且对任意的 $a\in R$，$\boldsymbol{M}_{R-\{a\}}$ 中必存在空集元素 $\varnothing$，则 R 是决策信息系统的一个约简。

基于特征矩阵的约简算法描述如算法 8-1 所示。

算法 8-1　基于特征矩阵的决策表属性约简算法

0	Algorithm：Attribute reduction based on characteristic matrix
1	输入决策表：$S=(U,C\cup D)$；
2	置决策表 S 的最小属性子集 Redu = $\varnothing$，$B=C$；
3	用信息增益公式计算各条件属性的信息增益，并取使信息增益最大的属性：$\text{Gain}(c)=\max\text{Gain}(a)$，$a\in C$，Redu = Redu $\cup\{c\}$，$B=C-\{c\}$；
4	计算决策表 S 关于条件属性 c 的特征矩阵 $\boldsymbol{M}_c$
5	若 $\boldsymbol{M}_c$ 无空集元素，则 c 即决策表的最小约简，结束；否则，求 B 中各属性相对于属性 c 的互补元素集合；
6	在 B 中选择相对于 $\boldsymbol{M}_c$（空集元素处）互补元素最多的属性 b，Redu = Redu $\cup\{b\}$，$B=B-\{b\}$；
7	若属性 b 的互补元素能覆盖 $\boldsymbol{M}_c$ 的空集元素，则结束；否则，在 $\boldsymbol{M}_c$ 中相应于 b 有关互补元素位置（此处 $\boldsymbol{M}_c$ 有空集元素）做上非空记号，转 Step 6。
8	输出约简：Redu；

2．基于属性重要性和频度的启发式约简算法

RedFreSigni 等提出基于属性重要性和频度的启发式约简算法（算法 8-2），该算法把用户偏好集同时作为属性近似约简的一部分，并以频度作为选择属性的启发信息。

给定决策信息系统 $S=(U,C\cup D)$，条件属性集 C 对决策属性集 D 的依赖程度为

$$r(C,D)=\frac{\left|\bigcup_{X\in D^*}C\_(X)\right|}{|U|} \tag{8-2}$$

其中，$D^*=U/D$，$C_(X)$为 X 的下近似。

给定决策信息系统 $S=(U,C\cup D)$，属性 $a\in C$ 的重要性为

$$\mathrm{SGF}(a,C,D)=r(C,D)-r(C,C\cup D) \tag{8-3}$$

对决策信息系统 $S=(U,C\cup D)$，用 $A=(U,R_C)$ 表示知识库，其中，R_C 表示由条件属性集 C 确定的等价关系，U/C 中的集合为原子集，原子集表示初等概念。$r(C,D)$ 的意义为，知识库 $A'=(U,R_D)$ 中能用知识库 $A=(U,R_C)$ 中的基本概念精确定义的基本概念所包含的对象数和 U 中对象数的比值。显然，知识库 $A=(U,R_C)$ 中的初等概念越精细，则上述精确程度越高。极端情况下，用条件属性集 C 不能对数据集 U 进行有意义的划分（将 U 的所有对象都归为一类），显然，只要决策属性集能对数据集进行有意义的划分（决策等价类个数大于 1），则 $r(C,D)=0$。相反，若用条件属性集 C 对数据集 U 划分的结果是每个对象都属不同的类，那么，知识库 $A'=(U,R_D)$ 中的每个初等概念都可用知识库 $A=(U,R_C)$ 中的基本概念来精确定义，所以，$r(C,D)=1$。若属性集 $V\subseteq C$，$r(C,D)=r(V,D)$，则表明用知识库 $A=(U,R_C)$ 中的基本概念精确定义 $A'=(U,R_D)$ 中的基本概念的能力和用 $A^*=(U,R_V)$ 中的基本概念精确定义 $A'=(U,R_D)$ 中的基本概念的能力相同。$r(C,D)$ 也可以用集合类关于近似空间的近似质量来理解，设 $F=U/D=\{Y_1,Y_2,\cdots,Y_n\}$，则 $r(C,D)=r_C(F)$。

算法 8-2　RedFreSigni 算法	
0	Algorithm：RedFreSigni Algorithm
1	输入决策表：$S=(U,C \cup D)$，用户偏好集 UP（用户认为比较重要的条件属性子集）
2	For （i=1；$i \leqslant n$；++i） count(c_i)=0；　　//其中 $c_i \in$C
3	M = DisMat(S)；count = CalFre(M)；
4	Core=GeneCore(M,count)；
5	Redu = Core∪UP；
6	AR = C-Redu；
7	sort(AR)；
9	depRed = r(Redu,D)；depC = r(C,D)；
10	while（depRed < depC）
10.1	c_i = selectmax(AR)；
10.2	RedOld = Redu；
10.3	Redu =Redu ∪{c_i}，AR=AR-{c_i}；
10.4	depRed = r(Redu,D)；depRedOld = r(depRedOld,D)；
10.5	if（depRed = depRedOld）
10.5.1	Redu = Redu－{ c_i }；
10.5.2	depRed = depRedOld； Endif
11	输出约简：Redu；

8.1.3　基于遗传算法的属性约简

下面介绍基于幂集演化的属性约简方法，其本质是遗传算法在属性约简中的应用，这里重点介绍针对属性约简特殊的编码机制、评价机制和遗传操作。

1．编码

遗传算法的第一步是初始化种群。首先，需要对问题进行遗传编码。编码就是将问题的解用一种码来表示，从而将问题的状态空间与 GA 的码空间相对应，这在很大程度上依赖于问题的性质，并将影响遗传操作的设计。由于 GA 的优化过程不是直接作用在问题参数本身，而是在一定编码机制对应的码空间上进行的，因此，编码的选择是影响算法性能与效率的主要问题。

本书采用幂集空间上的直接编码方式，不妨假设：问题的条件属性集 $C=\{p_1,p_2,\mathbf{L}\ p_n\}$，定义个体（染色体）$x\in S_C$，其中 S_C={{p_1},$\mathbf{L}$,{p_n},$\mathbf{L}$,{p_1,p_2},$\mathbf{L}$,{p_{n-1},p_n},$\mathbf{L}$,{p_1,p_2,p_3},$\mathbf{L}$,{p_1,$\mathbf{L}$,p_n}} 为 C 的幂集，算法初始时在空间 S_C 上随机的产生 k 个不同的个体，构成群体 $\mathrm{POP}=\{x_1,\mathbf{L},x_k\}$，每个个体 x_m 形如 $x_m=\{p_i,\mathbf{L},p_j\}$，$m=1,\mathbf{L},k;\ i,j=1,\mathbf{L},n$。

2．个体评价

在遗传算法中，适应度函数是用来判断群体中个体好坏的标准。对于属性约简来说，个体的适应度主要取决于两个方面：所含属性的个数（应尽量少）和区分能力（应尽量强）。由于采用幂集编码方式，这里的每一个个体 x 也可以看作条件属性 C 的一个子集，可以用 x 相对于 D 的条件熵刻画所含属性的区分能力，据此定义适应度函数：

$$f(x)=w(H(D)-H(D\mid x))+(1-w)(L(C)-L(x)) \qquad (8\text{-}4)$$

其中，$L(x)$ 为个体 x 的长度，也就是 x 中元素（属性）的个数，$L(C)$ 为整个条件属性集中属性的个数；w 为权重系数。

3．遗传操作

1）交叉

按如下方法构造幂集空间上的交叉算子 c，显然，$c:S_C\times S_C\to S_C\times S_C$。不妨假设 x_u、x_w 是群体 $\mathrm{POP}=\{x_1,\mathbf{L},x_k\}$ 中任意两个个体，即 $x_u,x_w\in\mathrm{POP}$，其中，x_u={p_{i1},p_{i2},p_{i3},$\mathbf{L}$,p_{iu}}，x_w={p_{j1},p_{j2},p_{j3},$\mathbf{L}$,p_{jw}}，且 $\{p_{i1},p_{i2},p_{i3},\mathbf{L},p_{iu}\}\ \mathbf{I}\ \{p_{j1},p_{j2},p_{j3},\mathbf{L},p_{jw}\}=\varnothing$，$c(x_u,x_w)$ 产生两个新的染色体 x_u'，x_w' 如下所示：

父代　　子代

$p_{i1}\mid p_{i2},p_{i3},\mathbf{L},p_{iu}$　　$p_{i1}\mid p_{j3},\mathbf{L},p_{jw}$

×

$p_{j1},p_{j2}\mid p_{j3},\mathbf{L},p_{jw}$　　$p_{j1},p_{j2}\mid p_{i2},p_{i3},\mathbf{L},p_{iu}$

当$\{p_{i1}, p_{i2}, p_{i3}, \mathbf{L}, p_{iu}\} \mathbf{I} \{p_{j1}, p_{j2}, p_{j3}, \mathbf{L}, p_{jw}\} \neq \varnothing$时，在子代个体$x_u^{'} = \{p_{i1}, p_{j3}, \mathbf{L}, p_{jw}\}$和$x_w^{'} = \{p_{j1}, p_{j2}, p_{i2}, p_{i3}, \mathbf{L}, p_{iu}\}$中去掉重复的属性。上述过程为单点交叉操作，同样可以构造出双点交叉等不同类型的交叉操作。

2）变异

幂集空间上的变异操作可以定义为算子$m: S_C \to S_C$，设个体x_u=$\{p_{i1}, p_{i2}, p_{i3}, \mathbf{L}, p_{iu}\}$，$p_{im} \in C$，$m$=1,2, $\mathbf{L}$,iu，变异算子m作用于x_u可以产生一个新的个体$x_u^{'}$=$\{p_{i1}, p_{i2}, \mathbf{L}, p_{il}^{'}, \mathbf{L}, p_{iu}\}$，其中，$l$是变异点，为[1,$u$]上的一个随机数。$p_{il}^{'} \in p - \{p_{i1}, p_{i2}, p_{i3}, \mathbf{L}, p_{iu}\}$，若$x_u$=$\{p_1, p_2, \mathbf{L}, p_n\}$，$m(x_u)$=$x_u$。

4．算法描述

在以上定义和操作的基础上，这里具体阐述基于幂集演化的属性约简方法，如算法8-3所示。

算法 8-3　幂集演化属性约简方法

```
0     Algorithm：Attribute reduction based on GA
1     t=0，initialize (POP)，evaluate(POP)
2     while（not terminate）
2.1       select（POP）;          //从 POP(t)中选择下一代 POP(t+1)
2.2       crossover（POP）;       /对 POP(t+1)进行杂交操作
2.3       mutate（POP）;          //对 POP(t+1)进行变异操作
2.4       evaluate（POP）;        //对 POP(t+1)进行评估
3     解码并输出最小约简
```

针对属性约简问题存在诸多方法，然而，到目前为止，还没有一个公认的、高效的最佳属性约简算法，研究者已经在理论上证明了求取最小约简是一个NP完全问题。幂集演化的属性约简方法作为一种基于遗传算法的属性约简方法，编码方式采用在属性集的幂集空间上进行直接编码，由于其编码具有不定长的特性，因此，在遗传操作策略上采取了一种与二进制编码完全不同的交叉和变异策略，使得该方法在选取合适评价函数的权重参数时，能明显加快算法的收敛速度。实验结果表明：一方面，该算法能对大多数测试数据集求得全部约简；另一方面，用该算法得到最小约简所用的平均进化代数小于其他方法。

8.2 电力负荷关联规则提取

8.2.1 问题概述

电力系统负荷预测的研究起源于 20 世纪初期，是指从已知的电力、经济、社会、气象等情况出发，通过对历史数据的分析和研究，探索事物之间的内在联系和发展变化规律，确定未来某特定时刻的电力需求量或用电量，对负荷发展做出预先估计和推测。目前，负荷预测是电力系统调度、规划等管理部门的重要工作之一，其预测水平已成为衡量一个电力企业管理是否走向现代化的标志。准确的负荷预测，有利于经济、合理地安排电网内部发电机组的启停，保证电网运行的安全，减少不必要的旋转储备容量；有利于用电管理和降低发电成本，提高电力系统的经济效益和社会效益；准确的负荷预测同时也是电源建设规划、安排电网增容和改建、决定电网的建设和发展的重要依据，其预测水平已成为衡量一个电力企业管理是否走向现代化管理的标志之一。

负荷预测按时间期限进行分类，通常可分为长期、中期、短期、超短期负荷预测。

1．长期负荷预测

长期负荷预测一般是指 10 年以上并以年为单位的预测，其目的是合理安排电源和电网的建设进度，提供宏观决策的依据，使电力建设满足国民经济增长和人民生活水平提高的需要。

2．中期预测

中期预测通常是指以年为单位的预测，它的预测结果通常用于指导燃料供应计划和机组维修计划的制订、电厂税额估计等。

3．短期预测

短期预测是指一年之内以月为单位的负荷预测，还指以周、天、小时为单位的负荷预测，通常预测未来一个月、未来一周、一天的负荷指标，也预测未来一天 24 小时中的负荷，多用于机组协调、火电协调、负荷管理等方面。

4．超短期负荷预测

超短期负荷预测是指未来半小时甚至未来 10 分钟的预测，其意义在于可对电网进行计算机在线控制，实现发电容量的合理调度，满足给定的运行要求，同时使发电成本最小。

对于不同类型预测，其所用的理论及研究方法有很大区别。其中，短期和超短期负荷预测的研究是最重要也是最困难的，提高短期负荷预测的精度既能增强电力系统运行的安全性，同时也能改善电力系统运行的经济性，已成为电力系统不容忽视的研究课题之一。

从 20 世纪 90 年代初开始，随着电力系统数据管理的日益规范，积累了大量有意义的数据，同时电力系统与气象等部门合作，获取了大量影响负荷变化的气象信息。因此，运用相关技术，从这些数据中提取出人们感兴趣的、隐含的、先前未知的、对决策有潜在价值的知识，已经成为可能。下面介绍采用 DPSO 解决电力负荷关联规则挖掘问题时，所采取的一些策略，这些思想对其他领域关联规则提取也具有一定的借鉴意义。

8.2.2　关联规则

1993 年，Agrawal 等人首次提出关联规则问题，并逐渐成为数据挖掘领域最重要的问题之一。下面对关联规则的概念和研究内容进行阐述。

在关联规则挖掘中，所谓的项目（Item）是指交易数据库中的一个属性字段，每个字段有一定的取值范围；包含若干个项目的集合被称为项集（Itemset）；把一个项集所包含的项目的个数称为此项集的维数或项集的长度。长度为 k 的项集，称作 k 维项集；在某个数据处理过程中，涉及的所有项目的集合称为事务（Transaction）。

定义 8-3　支持度（Support）：假定 X 是一个项集，D 是一个事务集合或交易数据库，称 D 中包含 X 的交易的个数与 D 中总的事务个数之比为 X 在 D 中的支持度。把 X 的支持度记作 $\sup(X)$，而关联规则 $X \rightarrow Y$ 的支持度则记作 $\sup(X \cup Y)$。

定义 8-4　可信度（Confidence）：对形如 $X \rightarrow Y$ 的关联规则，其中 X 和 Y 都是项集，定义规则的可信度为事务集合 D 中既包含 X、也包含 Y 的事务个数与 D 中仅包含 X 而不包含 Y 的事务个数之比，或者说是项集 $X \cup Y$ 的支持度与 X 支持关联规则挖掘的相关

工作之比，即 sup($X\cup Y$)/sup(X)。把规则 $X\rightarrow Y$ 的可信度记作 conf($X\rightarrow Y$)。事实上，可信度即指在出现了项集 X 的事务中，项集 Y 也同时出现的概率有多大。支持度和可信度都是规格化的概念，它们的范围都为 0～1。

定义 8-5　最小支持度（Minimum Support）：由用户定义的衡量支持度的一个阈，表示项集在统计意义上的最低重要性，记作 minsup。

定义 8-6　最小可信度（Minimum Confidence）：由用户定义的衡量可信度的一个值，表示规则的最低可靠性，记作 minconf。

可信度是对关联规则准确度的度量，或者说表示规则的强度；支持度是对关联规则重要性的度量，表示规则的频度。支持度说明了这条规则在所有事务中有多大的代表性，显然，支持度越大，关联规则越重要。有些关联规则可信度虽然高，但支持度却很低，说明该关联规则使用的机会很小。反之，如果支持度很高，可信度很低，则说明该规则不可靠。

定义 8-7　频繁项集（Frequent Itemset）：对一个项集 X，如果 X 的支持度小于用户定义的最小支持度阈值，即 sup(X)≥minsup，称 X 为频繁项集或集（Large Itemset）。

定义 8-8　非频繁项集（Not Frequent Itemset）：对一个项集 X，如果 X 的支度小于用户定义的最小支持度阈值，即 sup(X)<minsup，称 X 为非频繁项集或小集（Small Itemset）。

定义 8-9　最大频繁项集（Maximal Frequent Itemset）：若某频繁项集不是其他任何频繁项集的子集，则称之为最大频繁项集。

事实上，人们一般只对那些满足一定的支持和可信度的关联规则感兴趣。因此，为了发现有意义的关联规则，需要由用户给定两个基本阈值：最小支持度和最小可信度。关联规则的挖掘问题被归纳成如下两个子问题或两个基本步骤：

① 找到所有满足用户给定的最小支持度的频繁项集；

② 在频繁项集的基础上生成所有满足用户给定的最小可信度的关联规则。

关联规则是形如 $X \to Y$ 的规则，关联规则挖掘找出支持度和可信度分别大于或等于用户指定的最小支持度和最小可信度的关联规则。设 $I=\{i_1,i_2,\mathbf{L},i_m\}$ 是由 m 个不同项目组成的集合。给定一个事务数据库 D，其中，每一个事务 T 是 I 中的一组项目的集合，即 $T \subseteq I$，T 有一个唯一的标识符（TID），若项集 $X \subseteq I$ 且 $X \subseteq T$，则事务 T 包含项集 X，一条关联规则就是形如 $X \to Y$ 的蕴涵式，其中，$X \subseteq I$，$Y \subseteq I$，$X \mathbf{I} Y=f$，X 表示此关联规则的前件或前提（Antecedent），Y 为此关联规则的后件或结论（Consequent），关联规则 $X \to Y$ 成立的条件如下：

① 它具有支持度 s，即事务数据库 D 中至少有 s% 的事务包含 $X \mathbf{U} Y$；

② 它具有置信度 e，即事务数据库 D 中包含 X 的事务中至少有 e% 的事务包含 Y。

关联规则的采掘问题就是在事务数据库 D 中找出具有用户给定的最小支持度 minsup 和最小置信度 minconf 的关联规则。

由上文所述，关联规则的挖掘问题可以分解为以下两个问题：

① 找出存在于事务数据库中的所有大项集。项集 X 的支持度 support(X)不小于用户给定的最小支持度 minsup，则称 X 为大项集；

② 利用大项集生成关联规则。对于每个大项集 A，若 $B \subset A$，$B \neq f$ 且 $\dfrac{\text{support}(A)}{\text{support}(B)} \geqslant \text{minconf}$，则存在关联规则 $B \to (A-B)$。

在上述两个步骤中，第一个步骤集中了所有的计算量，通常称第一个子问题为经典关联规则挖掘问题。关联规则挖掘算法的性能主要由第一个问题决定，其主要是由数据量巨大所造成的，算法的效率及可扩展性都具有很强的挑战性，解决这一问题的最著名算法是 R.Agrawal 提出的 Apriori 算法，以及它的变种 Apriori-Tid 和 AprioriHybrid 算法。第二个步骤虽然很简单，但通常算法所返回的结果都非常庞大，而且还可能伴随着错误信息，如何从大量规则中找到有意义的规则，让用户更方便地解释和理解规则也非常重要。

自从关联规则挖掘问题提出以来，围绕经典关联规则挖掘算法和减少候选项集的生成数量，许多学者对关联规则的挖掘方法进行了研究，提出了许多关联规则挖掘算法，

这些算法大都围绕如何快速高效地生成频繁项集这一核心问题进行展开。根据频繁项集的表示方法不同，可以将其分为 3 种：完全频繁项集、频繁闭项集和最大频繁项集。下面对频繁项集的挖掘技术进行阐述。

8.2.3 频繁项集挖掘

关联规则反映了大量数据中项集之间的关联。从大量数据中发现满足最小支持度的频繁项集是关联规则挖掘的关键步骤。Apriori、AIS 和 Partition 等算法是频繁项集挖掘的代表，其中，以 Agrawal 等人提出的 Apriori 算法最为著名，其后的数据挖掘算法大多建立在 Apriori 算法基础之上，以 Apriori 为基础的算法基本上都采用了自底向上宽度优先的搜索策略，从一维频繁项集开始对数据库进行多次扫描，直到所有的频繁项集被发现，这些算法穷举每一个频繁项集，考虑对任何一个长度 L 的频繁项集 F，由于 F 的每个子集都是频繁项集合，这就是说算法必须产生 F 所有的 $2L$ 个子集，当 L 很大时，这将是 NP 难的问题。同样，如果在数据集中包含的不同的项目的数量为 n 个，则将要计算 $2n$ 个项集。当 n 比较大时，将会产生组合爆炸，实际上这也是 NP 难的问题。这些算法在读入交易数据时生成候选项集，产生许多不必要的候选项集，计算量大。尤其对海量数据集合来说，以上算法只有在较高的最小支持度和最小可信度下或增加其他约束后才有一定的挖掘效率，否则，将会产生频繁项集的组合爆炸，而变得效率低下，甚至超过机器的存储和计算能力。下面介绍几种挖掘频繁项集相关研究。

1．完全频繁项集挖掘

完全频繁项集的挖掘算法大致可以分为 3 种：广度优先搜索算法、深度优先搜索算法、广度和深度相结合的搜索算法。广度优先搜索算法的基本思想是自底向上地遍历整个问题空间，迭代地生成候选频繁项集，然后测试产生的项集是否为频繁项集。深度优先搜索算法与广度优先搜索算法不同，在深度优先搜索算法中，搜索会沿着搜索空间梯格的一条分支纵深向下，再回溯，迭代进行，直至计数完毕。广度优先算法和深度优先算法对于不同的数据特点具有独特的优势，因此，有学者考虑将两种搜索方式相结合，提出了广度和深度优先相结合的搜索算法。

2．频繁闭项集挖掘

频繁项集的数量与数据库中项目数呈指数关系，因此，挖掘完全频繁项集往往很困难，而且整个集合包含较多的冗余性。一些研究者提出频繁闭项集的概念，它是不被其他具有相同支持率的项集所包含的频繁项集，从中可抽取出无冗余的完全频繁项集。频繁闭项集提供了完全频繁项集的一种完整的、最小表示。它能够唯一确定完全频繁项集且规模小得多。由于频繁闭项集的数量要大大少于完全频繁项集，而且又具有能够重新计算其支持度的特点，挖掘频繁闭项集也日益成为一个新的研究热点。

3．最大频繁项集挖掘

虽然频繁闭项集在保留支持度信息的前提下极大地减少了频繁项集的数量，但当处理更稠密的数据集或最小支持度较少时，频繁闭项集的数目也会迅速增加，并且其中包含大量的冗余信息。在很多情况下，用户往往仅关心获取满足最小支持度的频繁项集，尽量减少项集间的冗余度。最大频繁项集的表示方法能够较好地解决此类问题。最大频繁项集是指那些在所有的频繁项集中不存在超集的频繁项集。由于最大频繁项集的个数小于频繁闭项集，更远小于完全频繁项集，所以，挖掘最大频繁项集可以有效缩小项集规模，因此得到了广泛的关注。只有从本质上减少生成不必要的项集，减少对项集的支持度的计算时间，才能大大减少频繁项集生成的数量和缩短数据挖掘的时间。由于任何频繁项集都是最大频繁项集的子集，所以，发现频繁项集的问题可以转化为发现所有最大频繁项集的问题，求取最大频繁项集的方法有 1998 年 Roberto J.Bayardo Jr 提出的 Max-Miner 算法，1998 年 Lin 等人提出的 Pincer-Search 算法等。Max-Miner 算法根据不同项目之间的顺序，采用集合枚举树来描述项集，它突破了传统的自底向上的搜索策略，采用自底向上和自顶向下的搜索策略同时进行搜索，提出向前看（Look Ahead）的剪枝策略，尽可能早地对项集进行修剪，使最大频繁项集发现过程转化为在集合枚举树的搜索过程，虽然可以避免一些候选最大频繁项集的生成，但并没有充分利用在剪枝时生成的非频繁项集信息，仍然产生许多无用的候选最大频繁项集。Pincer-Search 算法采用了自底向上和自顶向下的双向搜索策略，但其第 k 次的 MFCS 是由 k-1 次的 MFCS 中的非

频繁项集去掉一个元素来生成的，产生了过多的无用候选项集，对海量数据库来讲，算法 Pincer-Search 将陷入 NP 难度的陷阱。

8.2.4 基于 DPSO 方法负荷规则萃取

1．负荷区间划分

以某省负荷数据为例，考查该省单日 96 点负荷曲线，总体上表现出如下特征：曲线从午夜零时到凌晨 4 时相对稳定，这一阶段的负荷最低点通常出现在 4 时左右，并以此为拐点，负荷逐渐增大，到上午 7 时，负荷开始减小，直到上午 8 时，负荷重新增长，在上午 11 时达到这一阶段的最高峰。上午 11 时到下午 13 时为午休时间，在这一时间段上负荷值减小，从下午 13 时开始，负荷又逐渐增大，到下午 18 时左右负荷逐渐达到一个新的高点，在冬季这个高点会有些提前，这个高点之后，负荷值逐渐减小，期间在 19 时左右，有 1 个小的反弹。事实上，气象因素对不同阶段的影响是不同的，因此，将 96 点数据划分成几个区间。应该强调的是，这种划分是人为的，区间划分的太粗或太细都是不合适的，如果分得过细，以极端的情况为例，把每个数据点看成一个区间，这样会使一种气象条件蕴含了 96 条规则，规则的可信性降低。换句话说，无法得出某种气象条件对某一时刻的影响情况；反过来，区间划分过粗，如将全部 96 点看成一个区间，这样划分有利于规则的萃取，但得到的规则过于粗糙，规则对 96 点数据整体有较高的可信性，但在局部上是不够准确的。这时不得不在这两种情况之间做折中的考虑，并根据日常生活工作情况的时段划分，将一天的负荷数据分成 4 个阶段，记为 G_1、G_2、G_3、G_4，如图 8-1 所示。

其中，G_1 为零时到上午 8 时，G_2 为 8 时到下午 13 时，G_3 为下午 13 时到下午 18 时，G_4 为下午 18 时到次日零时。

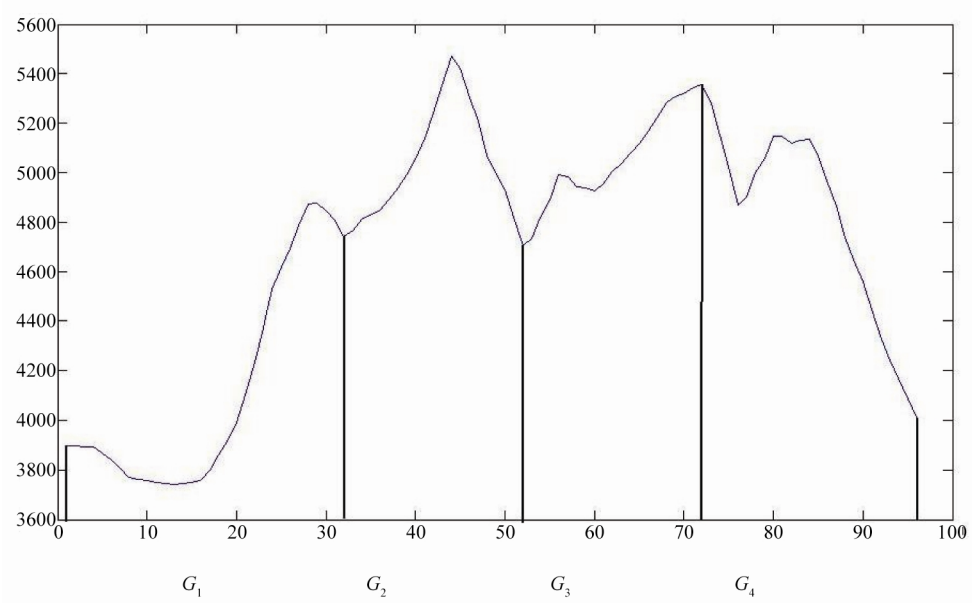


图 8-1　负荷区间划分

2．粒子的表示

将不同阶段的负荷数据融入实感温度形成样本集。在样本集中，气象等影响因素的变化决定了负荷曲线的变化，每一个样本都确定了一条规则 $X \to Y$，其中，X 为实感温度、日期类型等因素，称为条件属性，Y 是与之对应的负荷值，称为决策属性。在负荷预测的过程中，气象因素的累积效应是需要考虑的，这里考虑了前 3 天气象变化对负荷的影响，将前 3 天和当前的实感温度作为条件属性，由于要考查的是相关气象因素的变化对负荷的影响，在这里决策属性并不是负荷本身，而是当前负荷值与前一天负荷值变化的百分比，具体可通过下式计算：

$$Y_j^{(i)} = \frac{L_j^{(i)} - L_{j-1}^{(i)}}{L_{j-1}^{(i)}} \tag{8-5}$$

其中，$i = 1, \mathbf{L}, 4$ 为图 8-1 所示的负荷区间划分的个数，$j = 1, \mathbf{L}, m$，m 为集中样本的个数，$L_j^{(i)}$、$L_{j-1}^{(i)}$ 为第 j 和 $j-1$ 条 96 点数据第 i 个区间负荷的平均值，$Y_j^{(i)}$ 为当前负荷值与前一天负荷值变化的百分比。

将条件属性离散化，设 R 为一条件属性，W 为属性 R 所有取值的集合。按实际需要将 W 划分成 N 个互不相交的子集，即 $\bigcup_{n=1}^{N} U_n = W$，$U_i \cap U_j = \Phi, i \neq j$。以此为标准将属性 R 拆分成 N 个属性，如图 8-2 所示。

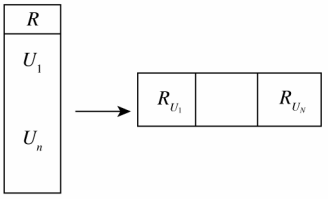


图 8-2　条件属性的离散

按实际需要，将决策属性分成若干类，条件属性与这些类别相对应，进而形成新样本集。

在 DPSO 方法中，粒子的表示方式与具体的问题密切相关。不妨设 $T = (X_1, \mathbf{L}, X_N, Y_1, \mathbf{L}, Y_M)$ 为一个样本集，其中，$X_i (i = 1, \mathbf{L}, N)$ 为条件属性，$Y_j (j = 1, \mathbf{L}, M)$ 为决策属性，且有 $X_1 \wedge X_2 \wedge \mathbf{L} \wedge X_N \rightarrow Y_1, Y_2, \mathbf{L}, Y_M$。假设对于任意条件属性 X_i 和决策属性 Y_j，分别有 S_i 和 T_j 种不同取值，对于任意条件属性 X_i，假设离散后 X_i 被划成 M 个互不相交的子集，用 M 位二进制编码即可对 X_i 进行表示，称这种刻画一个条件属性的二进制串为基因片段，用 $B(X_i)$ 表示，用一个长度为 $\sum (S_i + T_j)$ 的二进制串来表示 $X_1 \wedge X_2 \wedge \mathbf{L} \wedge X_N \rightarrow Y_1, Y_2, \mathbf{L}, Y_M$，在这个二进制串中，第 $\sum_{k=1}^{i-1} (S_k) \sim \sum_{k=1}^{i} (S_k)$ 位之间的位串表示的是属性 X_i 的取值，由于 X_i 只能有一种取值，因此，$B(X_i)$ 中只能有一位，其值是 1，其他位全部为 0。每个粒子的具体形式如下：

$$X_1 \wedge X_2 \wedge \mathbf{L}\ X_i\ \mathbf{L} \wedge X_n \rightarrow Y$$
$$001\mathbf{L}\underbrace{0\mathbf{L}0}_{M}\mathbf{L}\,0\,\mathbf{L}\,010$$

例如，在实感温度为 20～28℃时，感觉较为舒适，因此，将实感温度分成 7 段，分别为：(-∞, -20]、(-20, -10]、(-10, 0]、(0, -10]、(10, 23]、(23, 28]、(28, +∞)。若实感

温度为 25℃，则该编码片段为 0000010。所有属性的编码方式是相同的，编码后的样本为一串长度相同的二进制串。

与条件属性一样，决策属性也采用这种编码方式，并置于条件属性之后。表示决策属性的编码的长度与决策属性的分类数有关。

为便于叙述，用 X 表示 $X_1 \wedge X_2 \wedge \mathbf{L} \wedge X_n$，设每个样本都确定的相应规则为 $X \to Y$，令

$$\begin{aligned} T &= |X \,\&\, Y| \\ F &= |\neg X \,\&\, Y| \\ N &= |X \,\&\, \neg Y| \end{aligned} \tag{8-6}$$

其中，T 为样本集中满足规则 $X \to Y$ 的样本个数；F 为样本集中满足规则 $\neg X \to Y$ 的样本个数；N 为样本集中满足规则 $X \to \neg Y$ 的样本个数。定义评价函数

$$\text{Fitness} = \frac{T}{(T+F)} \cdot \frac{T}{(T+N)} \tag{8-7}$$

由式（8-7）可知，评价函数由两部分组成，其中，$\frac{T}{(T+F)}$ 通过决策属性对 $X \to Y$ 进行估价，反映的是条件属性为 X 且决策属性为 Y 的样本与所有决策属性为 Y 的样本比例；$\frac{T}{(T+N)}$ 则从条件属性出发，反映的是所有条件属性为 X 且决策属性为 Y 的样本与所有条件属性为 X 的样本的比例。

由于对粒子的表达方式进行了特殊的规定，粒子的修正策略也要进行相应的调整。不妨设任意属性 X_i 存在 M 个取值，记为 $V_1, V_2, \mathbf{L}, V_M$，用 c_i 表示在数据集 T 中，属性 X_i 的值为 V_i 的记录的个数，$B(X_i)$ 中存在两个或两个以上 1，则根据概率 $P = \frac{c_i}{\sum c_i}$ 来决定哪一位的值为 1，而在 $B(X_i)$ 的其他位置上的值全部为 0。通过这种策略以保证潜在解有意义。

采用上述方法获取了反映实感温度与负荷变化之间规律的规则，由于实感温度本身是由气温、风速、湿度等气象因素计算而来，故上述规则建立起了这些因素与负荷变化之间的联系。在进行负荷预测时，首先，根据传统的“近大远小”模式生成初步预测曲

线。然后，使用相应规则对初步预测曲线进行修正，由于规则中变化比率是一个范围，所以，通常采用这个区间的中心点作为修订值。

8.3 神经网络训练

人脑是由大量的生物神经元经过相互连接而成的一种高度复杂的、非线性的、并行的系统。因此，人们自然想到从模仿人脑智能的角度出发，来探求新的信息表示、存储和处理方式。人工神经网络（Artificial Neural Network，ANN）是仿效生物处理模式以获得智能信息处理功能的方法。神经网络着眼于脑的微观网络结构，通过大量神经元的复杂连接，由底至顶，通过自学习、自组织和非线性动力学所形成的并行分布方式，来处理难以语言化的模式信息。目前，人们对大脑的神经网络结构、运行机制甚至单个神经细胞的工作原理的了解还很肤浅，但基于生物神经系统的分布式存储、并行处理、自适应学习等现象，已经构造出具有一定的初级智能的人工神经网络，并在一些科研和实际应用中显示出很大的威力。实质上，人工神经网络是一个不依赖于模型的自适应函数估计器，因而，不需要模型就可以实现任意的函数关系。其突出的优点是能够并行处理，并具有学习能力、适应能力和很强的容错能力。神经网络在信息处理中的运用越来越广泛，特别是在模式识别、图像处理、预测、预报、人工智能、优化和控制等领域，已取得了令人瞩目的发展。

人工神经网络首先要以一定的学习准则进行学习，然后才能工作。神经网络的知识不是保存在存储器中，而是由组成神经网络的各个神经元之间的连接和连接强度来保存。网络上的每个神经元只记载少量的信息，它们只对输入数据进行简单的逻辑运算或算术运算，并把结果送到有关的连接上。知识的更新或网络的学习就是对神经元之间的连接和连接强度（权值）的修正。本节将对采用 PSO 方法对神经网络进行训练的方式及相关知识进行简单介绍。

8.3.1　神经元模型

到目前为止，人们已经建立了数百种人工神经元模型，最常用的神经元模型仍然是最早提出的 MP 模型。神经元是一个多输入、单输出的信息处理单元，神经网络就是由多个神经元加权连接而成的网络。虽然单个神经元只能进行十分简单的信息处理，但多个神经元连接而成的网络却具有强大的计算能力。神经网络计算表现为神经元之间的相互作用。改变神经元之间的连接方式和连接强度就可以改变神经网络的计算效果，其中，两个神经元之间的连接强度大小由一个实数表示，该实数称为连接权值。神经元之间的连接形式和连接权值通常由神经网络学习过程决定，根据神经元类型、神经元连接方式和学习方式不同，设计形成了各种不同的神经网络模型。

人工神经元是对生物神经元的简化和模拟，如图 8-3 所示为一个简化的神经元结构，它是一个多输入、单输出的非线性元件，是人工神经网络的基本处理单元，其输入/输出关系可描述为

$$O = f\left(\sum_{j=1}^{R} w_{ji}x_j - q_i\right) \tag{8-8}$$

其中，x_j（j=1,2, **L** , n）代表从其他细胞传来的输入信号；q_i 代表神经元的阈值；w_{ji} 代表从细胞 j 到细胞 i 的连接权值（对于激活状态，w_{ji} 取正值；对于抑制状态，w_{ji} 取负值）；O 为神经元输出；f(・)称为转移函数。

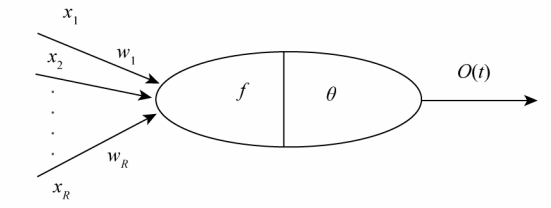


图 8-3　神经元模型

常用的转移函数有如下两个。

①　当 O 取 0 或 1 时，一般采用阶跃函数 $y = f(x) = \begin{cases} 1, x \geqslant 0 \\ 0, x < 0 \end{cases}$。

② 当O在(0,1)或(-1,1)内连续取值时，一般采用 Sigmoid 函数$y=f(x)=\frac{1}{1+e^{-bx}}$，其中，参数$b$用来控制曲线斜率，或采用双曲正切函数$y=f(x)=\frac{1-e^{-x}}{1+e^{-x}}$。

8.3.2 神经网络

由大量神经元按照某种方式进行连接就构成了神经网络，神经网络本身可以看作一个黑箱式的信息处理系统，如图 8-4 所示。每当网络接收到外部信息时，便输出一个经过加工处理后的结果。人可以通过修改网络连接方式和连接权值，使网络具有某种特殊的信息处理能力，但却不能控制网络中的每一个计算步骤。

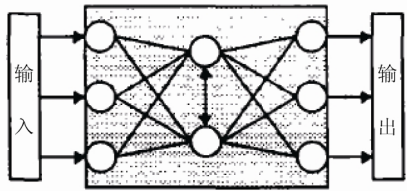


图 8-4 神经网络信息处理示意

研究神经网络的信息处理过程，则会发现神经网络有如下计算特征：

① 每个神经元独立于其他神经元进行工作，且每个神经元在某时刻的输出信息只依赖于那个时刻来自与其连接的其他神经元的有效信息；

② 每个神经元只能对来自局部的信息进行加工处理，一个神经元只能处理与其连接的神经元发送来的信息。

这些特征使神经网络具备了大规模并行计算能力和很好的容错能力，神经网络是由大量的神经元广泛互连而成的网络。根据连接方式的不同，神经网络的拓扑结构可分如下两大类。

1．**前向型网络**（Feedforward Network）

前向型网络由输入层、隐层和输出层组成，隐层可有若干层，每一层的神经元只接收其前一层神经元的输出，任何一个神经元可以有任意多个输入，但只有一个输出，

而输出可以耦合到任意多个其他结点的输入。前向型网络是应用最广泛的人工神经网络模型。

2．反馈型网络（Feedback Network）

反馈型网络的层间神经元结点，以及层内神经元结点间存在反馈连接，根据反馈方式的不同，反馈型网络又可分为从输出层到输入层有反馈的前向网络、层内有反馈的前向网络和相互结合型网络等。

8.3.3　神经网络的学习

人工神经网络的学习方式可分以下 3 种。

1．监督学习（有教师指导的学习）

监督学习即存在一组已知的输入/输出数据，网络根据已知输出和实际输出的差值（误差信号）来调节网络连接和连接权值。

2．无监督学习（无教师指导的学习）

无监督学习即网络根据外部环境提供的数据中的某些统计规律（如聚类特征或某些统计分布特征）来调节网络连接和连接权值。

3．增强学习

增强学习即外部环境只对网络输出结果给出评价（奖或惩），而不是给出正确答案，网络通过增强那些受奖励的动作和减弱那些受惩罚的动作来调节网络连接和连接权值。

下面介绍两种典型的学习算法。

（1）d 学习。d 学习是一种有教师指导的学习。设输入为 $x_1, x_2, \mathbf{L}$ ，x_n 时神经元 k 的实际输出为 y_k，而其应该的输出为 t_k，令误差信号 $d_k = t_k - y_k$，那么，根据梯度下降优化算法可得权值的修正为 $\Delta w_{jk} = h d_k x_j$，其中，$j = 1, 2, \mathbf{L}, n$；$h$ 是学习步长。d 学习仅用于单层网络的学习。

（2）Hebb 学习。Hebb 学习是一种无教师指导的学习。神经心理学家 Hebb 提出的学习规则可归结为“当连接神经元两端的激励信号同为激活或同为抑制时该连接的强度应增强，反之则减弱”。其数学公式可描述为：$\Delta w_{jk} = h y_k x_j$，其中，$h$ 是学习步长。

其他还有竞争（Competitive）学习、*Q* 学习和自适应启发式评判学习等。

8.3.4 前向神经网络

前向神经网络（Feedforward Neural Network）是应用最广泛的一种神经网络模型，它由一个输入层、一个输出层、一个或多个隐层构成，各层次的神经元之间单向全互联联接，是一种由非线性变换单元组成的前馈型网络，在模式识别、非线性预测和非线性函数逼近等领域获得了成功，下面对前向神经网络的逼近能力和训练过程进行简要阐述。

1．前向神经网络的函数逼近能力

一个 3 层前向神经网络的拓扑结构如图 8-5 所示，假设有 n 个输入层结点、n_1 个隐层结点、m 个输出层结点，各层结点的输出为 $Y_j^1 = f(\sum_{i=1}^{n} w_{ij} x_i - q_j), j = 1, 2, \mathbf{L}, n_1$。

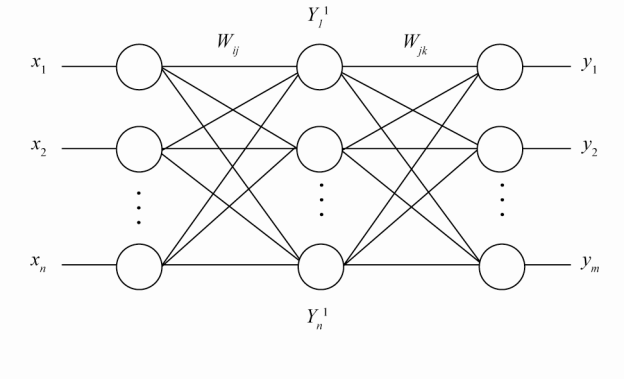


图 8-5　前向神经网络

理论研究已经证明，假定隐层结点可以根据需要自由设置，采用 *S* 状转移函数（Sigmoid 函数或双曲正切函数）的前向神经网络可以任意精度逼近任何连续函数，采用阶梯函数为转移函数的前向神经网络可以实现任意二值逻辑函数。

2．学习算法

设网络的神经元转移函数采用连续可导函数。有 N 个学习样本 $(\bar{X}_k,\bar{T}_k),k=1,2,\mathbf{L},N$，对于某个输入 $\bar{X}_k$ 网络的实际输出为 $\bar{Y}_k$，期望输出为 $\bar{T}$。令网络中某一结点 i 的输出为 Y_{ik}，i 和 j 的连接权值为 w_{ij}，那么结点 j 的输入加权和为 $\text{net}_{jk}=\sum_i w_{ij}Y_{ik}$。误差函数采用最小二乘误差函数 $E=\frac{1}{2}\sum_{k=1}^{N}(\bar{T}_k-\bar{Y})^2$，$E_k=(\bar{T}_k-\bar{Y}_k)^2=\sum_j(T_{jk}-Y_{jk})^2$。记 $d_{jk}=\frac{\partial E_k}{\partial \text{net}_{jk}}$，$O_{jk}=f(\text{net}_{jk})$，从而：

$$\frac{\partial E_k}{\partial w_{ij}}=\frac{\partial E_k}{\partial \text{net}_{jk}}\frac{\partial \text{net}_{jk}}{\partial w_{ij}}=\frac{\partial E_k}{\partial \text{net}_{jk}}O_{ik}=d_{jk}O_{ik} \tag{8-9}$$

① 当 j 为输出结点时，$O_{jk}=Y_{jk}$，有

$$d_{jk}=\frac{\partial E_k}{\partial \text{net}_{jk}}=\frac{\partial E_k}{\partial Y_{jk}}\frac{\partial Y_{jk}}{\partial \text{net}_{jk}}=-2(T_{jk}-Y_{jk})f'(\text{net}_{jk}) \tag{8-10}$$

② 当 j 不是输出结点时，$d_{jk}=\frac{\partial E_k}{\partial \text{net}_{jk}}=\frac{\partial E_k}{\partial O_{jk}}\frac{\partial O_{jk}}{\partial \text{net}_{jk}}=\frac{\partial E_k}{\partial O_{jk}}f'(\text{net}_{jk})$，有

$$\frac{\partial E_k}{\partial O_{jk}}=\sum_m\frac{\partial E_k}{\partial \text{net}_{jk}}\frac{\partial \text{net}_{jk}}{\partial O_{jk}}=\sum_m\frac{\partial E_k}{\partial \text{net}_{jk}}\frac{\partial(\sum_j w_{jm}O_{jk})}{\partial O_{jk}}=\sum_m d_{mk}w_{jm} \tag{8-11}$$

其中，$d_{jk}=f'(\text{net}_{jk})\sum_m d_{mk}w_{jm}$。

权值的修正量应取误差函数 E_k 对权值 w_{ij} 的负梯度：

$$w_{ij}(t+1)=w_{ij}(t)-h\sum_{k=1}^{N}\frac{\partial E_k}{\partial w_{ij}}=w_{ij}(t)-h\sum_{k=1}^{N}d_{jk}O_{ik} \tag{8-12}$$

其中，t 表示迭代次数，h 表示学习步长。学习过程由信号的正向传播与误差的反向传播两个过程组成。正向传播时，输入样本从输入层传入，经各隐层逐层处理后，传向输出层。若输出层的实际输出与期望的输出（教师信号）不符，则转入误差的反向传播阶段。误差反向传播是将输出误差以某种形式通过隐层向输入层逐层反传，并将误差分摊给各层的所有单元，从而获得各层单元的误差信号，此误差信号即作为修正各单元权值的依据。这种信号正向传播与误差反向传播的各层权值调整过程，是周而复始地进行的。

权值不断调整的过程，也就是网络的学习训练过程。此过程一直进行到网络输出的误差减少到可接受的程度，或进行到预先设定的学习次数为止。

3．存在的问题

由于前向网络是一个非常复杂的高维非线性系统，其理论分析困难，在实践中主要存在如下问题。

① 确定网络规模，即针对具体的问题，网络规模结构应如何设计？包括如何设计网络层数、各层的神经元数，以及神经元之间的连接。

② 学习效率问题。当网络规模较大时，学习训练过程往往很慢，且容易陷入局部极小点。

③ 权值初始化问题。基于梯度的学习算法对网络的初始值敏感，不适当的网络初始值易使网络的学习训练陷入局部极小点。

4．基于 PSO 前向网络学习

前向神经网络在诸多领域取得了巨大的成功，但是它有可能陷入局部最小值，不能保证收敛到全局极小点。另外，反向传播算法训练次数多，收敛速度慢，使学习结果不能令人满意。由于 BP 神经网络的学习过程主要是权值和阈值的更新过程，因而，可用 PSO 算法中粒子的位置来对应其全部连接权值和阈值，以神经网络的输出误差作为 PSO 算法的适应函数，通过 PSO 算法的优化搜索来训练神经网络的权值和阈值，以获得尽可能低的训练误差。

向前神经网络实现的是一种非线性映射，由一权值向量所确定，记作 $(w_1, w_2, \mathbf{L}, w_n)$。可将该向量看成 PSO 中的粒子，为方便起见，这里称之为神经粒子，用 NP 表示。因此每个神经粒子的输入、输出、隐层数及每个隐层的神经元个数是确定的，神经粒子的输出用 O(NP)表示。假设参与神经粒子学习的样本有 M 个，用 T_i 表示样本 i 的教师信号。用函数

$$F(\mathrm{NP}) = \sum_{i=1}^{M} \left| O(\mathrm{NP}) - T_i \right| \qquad (8\text{-}13)$$

来评价每个神经粒子的优劣。通过神经粒子在空间内搜索，而取代基于梯度的神经网络训练，算法 8-4 详细描述了神经粒子群方法。

算法 8-4　神经粒子群方法	
0	Algorithm：Neural particle swarm optimization
1	初始化神经粒子群 POP，假设 POP 由 N 个神经粒子构成，确定神经粒子的输入、神经元个数、激励函数等，在规定范围内为每个神经粒子随机生成一组权值；
2	根据一组学习样本和教师信号，采用式（8-13）对每个神经粒子 NP_j（j=1,2, $\mathbf{L}$,N）进行评价，记该粒子经过的最好位置为 P_NP_j，整个群体经过的最好位置为 G_NP_j；
3	按式（4-1）修正每一个粒子下一步位置；
4	若满足终止条件，则选出评价函数值最好的神经粒子，算法结束，否则，转到 Step2；

正如上文所述，除理论基础薄弱外，制约前馈型神经网络进一步发展的两个主要问题是：一是前馈型神经网络的学习算法收敛很慢，网络学习效率很低；二是前馈型神经网络对初值的选择非常敏感，不恰当的初值选择往往会造成网络的不收敛。由算法 8-4 可以看出，神经粒子群算法中粒子根据自身经验和群体经验来确定自身位置，没有十分复杂的数学运算，因此，效率得到很大的提高；同时，粒子分布在整个空间内，从而避免了初值选择对网络收敛性的影响。

8.4　小结

演化计算方法是一种通用的问题求解方法，其本质是将具体问题转化成解空间的搜索问题，具有自组织、自适应、自学习性和本质并行性等特点，采用演化计算方法求解现实问题有如下优势：

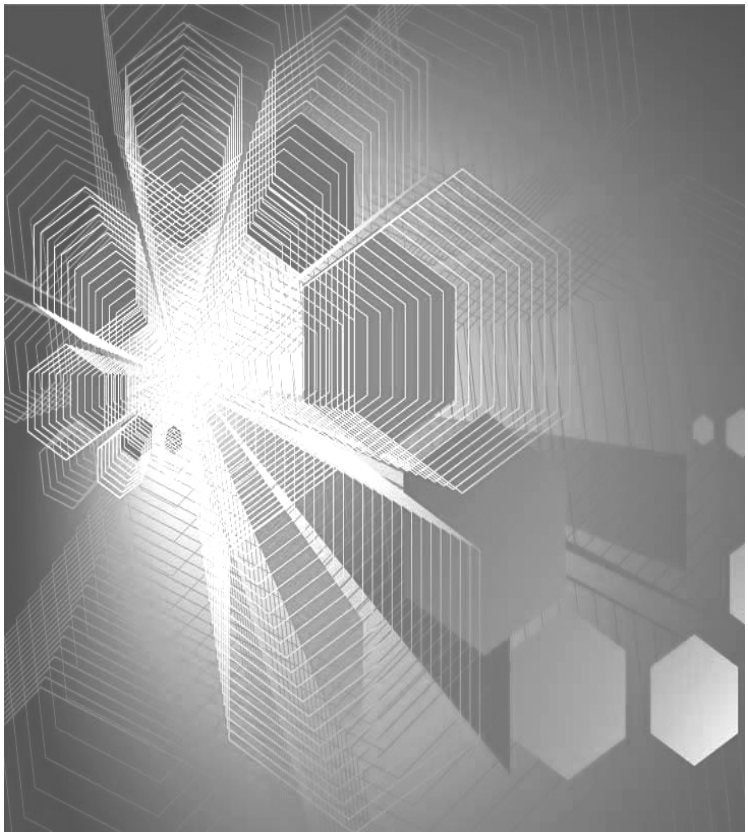
① 演化计算方法基于群体操作，因此，求取的结果基本不依赖初值的选取；

② 对搜索空间和启发信息没有特殊的要求；

③ 计算相对简单。

基于上述特点，演化计算方法越来越受到人们的青睐，并在大型优化问题求解、机器学习、自适应控制、人工生命、神经网络、经济预测等领域取得了成功，引起了包括数学、物理学、化学、生物学、计算机科学、社会科学、经济学及工程应用等领域科学们的极大兴趣。本书对遗传算法、进化规划、粒子群优化、微分演化和文化算法等演化类算法进行了详尽的阐述，并介绍了几个应用示例，对演化计算方法在工程实践中的应用，有较强的启发和参考意义。

附录 A　无约束优化问题



f_1：Sphere Model

$$f_1(x)=\sum_{i=1}^{n}x_i^2$$

$$-100\leqslant x_i\leqslant 100 \qquad \min(f_1)=f_1(0,\mathbf{L},0)=0$$

f_2：Schwefel's Problem Ⅰ

$$f_2(x)=\sum_{i=1}^{n}|x_i|+\prod_{i=1}^{n}|x_i|$$

$$-10\leqslant x_i\leqslant 10 \qquad \min(f_2)=f_2(0,\mathbf{L},0)=0$$

f_3：Schwefel's Problem Ⅱ

$$f_3(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$$

$$-100\leqslant x_i\leqslant 100 \qquad \min(f_3)=f_3(0,\mathbf{L},0)=0$$

f_4：Schwefel's Problem Ⅲ

$$f_4(x)=\max_i\{|x_i|,1\leqslant i\leqslant n\}$$

$$-100\leqslant x_i\leqslant 100 \qquad \min(f_4)=f_4(0,\mathbf{L},0)=0$$

f_5：Generalized Rosenbrock's Function

$$f_5(x)=\sum_{i=1}^{n}\left[100\left(x_{i+1}-x_i^2\right)^2+(x_i-1)^2\right]$$

$$-30\leqslant x_i\leqslant 30 \qquad \min(f_5)=f_5(1,\mathbf{L},1)=0$$

f_6：Step Function

$$f_6(x)=\sum_{i=1}^{n}\left(\lfloor x_i+0.5\rfloor\right)^2$$

$$-100\leqslant x_i\leqslant 100 \qquad \min(f_6)=f_6(0,\mathbf{L},0)=0$$

f_7：Quartic Function with Noise

$$f_7(x)=\sum_{i=1}^{n} ix_i^4+\text{random}[0,1)$$

$$-100\leqslant x_i\leqslant 100 \qquad \min(f_7)=f_7(0,\mathbf{L},0)=0$$

f_8：Generalized Schwelfel's Problem

$$f_8(x)=\sum_{i=1}^{n}\left(-x_i\sin\left(\sqrt{|x_i|}\right)\right)$$

$$-500\leqslant x_i\leqslant 500 \qquad \min(f_8)=f_8(420.9687,\mathbf{L},420.96870)=-418.9829n$$

f_9：Generalized Rastrigin's Function

$$f_9(x)=\sum_{i=1}^{n}\left[x_i^2-10\cos(2\pi x_i)+10\right]$$

$$-5.12\leqslant x_i\leqslant 5.12 \qquad \min(f_9)=f_9(0,\mathbf{L},0)=0$$

f_{10}：Ackley's Function

$$f_{10}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)+20-\mathrm{e}$$

$$-32\leqslant x_i\leqslant 32 \qquad \min(f_{10})=f_{10}(0,\mathbf{L},0)=0$$

f_{11}：Generalized Griewank Function

$$f_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$$

$$-600\leqslant x_i\leqslant 600 \qquad \min(f_{11})=f_{11}(0,\mathbf{L},0)=0$$

f_{12}：Generalized Penalized Function I

$$f_{12}(x)=\frac{\pi}{n}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$$

$$-50\leqslant x_i\leqslant 50 \qquad \min(f_{12})=f_{12}(1,\mathbf{L},1)=0$$

f_{13}：Generalized Penalized Function II

$$f_{13}(x)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right]+(x_i-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\}+\sum_{i=1}^{n}u(x_i,5,100,4)$$

$$-50\leqslant x_i\leqslant 50 \qquad \min(f_{13})=f_{13}(1,\mathbf{L},1)=0$$

其中，

$$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m & ,\ x_i>a\\ 0 & ,\ -a\leqslant x_i\leqslant a\\ k(-x_i-a)^m & ,\ x_i<-a\end{cases}$$

$$y_i=1+\frac{1}{4}(x_i+1)$$

f_{14}：Shekel's Foxholes Function

$$f_{14}(x)=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$$

$$-65.536\leqslant x_i\leqslant 65.536 \qquad \min(f_{14})=f_{14}(-32,-32)\approx 1$$

其中，

$$(a_{ij})=\begin{pmatrix}-32 & -16 & 0 & 16 & 32 & -32 & \mathbf{L} & 0 & 16 & 32\\ -32 & -32 & -32 & -32 & -32 & -16 & \mathbf{L} & 32 & 32 & 32\end{pmatrix}$$

f_{15}：Kowalik's Function

$$f_{15}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i+b_ix_3+x_4}\right]^2$$

$$-5\leqslant x_i\leqslant 5 \quad \min(f_{15})\approx f_{15}(0.1928,0.1908,0.1231,0.1358)\approx 0.0003075$$

其中，

i	a_i	b_i^{-1}
1	0.1957	0.25
2	0.1947	0.5
3	0.1735	1
4	0.1600	2
5	0.0844	4
6	0.0627	6
7	0.0456	8
8	0.0342	10
9	0.0323	12
10	0.0235	14
11	0.0246	16

f_{16}：Six-hump Camel-Back Function

$$f_{16}(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$$

$$-5\leqslant x_i\leqslant 5$$

$$\min(f_{16})=f_{16}(0.08983,-0.7126)=f_{16}(0.08983,0.7126)=-1.0316285$$

f_{17}：Branin Function

$$f_{17}(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos x_1+10$$

$$-5\leqslant x_1\leqslant 10,\ -5\leqslant x_1\leqslant 10$$

$$\min(f_{17})=f_{17}(-3.142,12.275)=f_{17}(-3.142,2.275)=f_{17}(9.425,2.425)=0.398$$

f_{18}：Goldstein-Price Function

$$f_{18}(x)=\left[1+(x_1+x_2+1)^2\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]\times$$
$$\left[30+(2x_1-3x_2)^2\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]$$

$$-2\leqslant x_i\leqslant 2\qquad \min(f_{18})=f_{18}(0,-1)=3$$

$f_{19} \sim f_{20}$：Hartman's Family

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{n} a_{ij}\left(x_j - p_{ij}\right)^2\right]$$

$$0 \leqslant x_i \leqslant 1$$

其中，对于 f_{19}：$n=3$，有

i	a_{ij}, j=1,2,3			c_i	p_{ij}, j=1,2,3		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.038150	0.5743	0.8828

$$\min\left(f_{19}\right) = f_{19}\left(0.114, 0.556, 0.852\right) = -3.86$$

对于 f_{20}：$n=6$，有

i	a_{ij}, j=1,⋯,6						c_i	p_{ij}, j=1,⋯,6					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

$$\min\left(f_{20}\right) = f_{20}\left(0.201, 0.150, 0.477, 0.275, 0.311, 0.657\right) = -3.32$$

$f_{21} \sim f_{23}$：Shekel's Family

$$f(x) = \sum_{i=1}^{m}\left[\left(x - a_i\right)\left(x - a_i\right)^{\mathrm{T}} + c_i\right]^{-1}$$

$$0 \leqslant x_i \leqslant 10$$

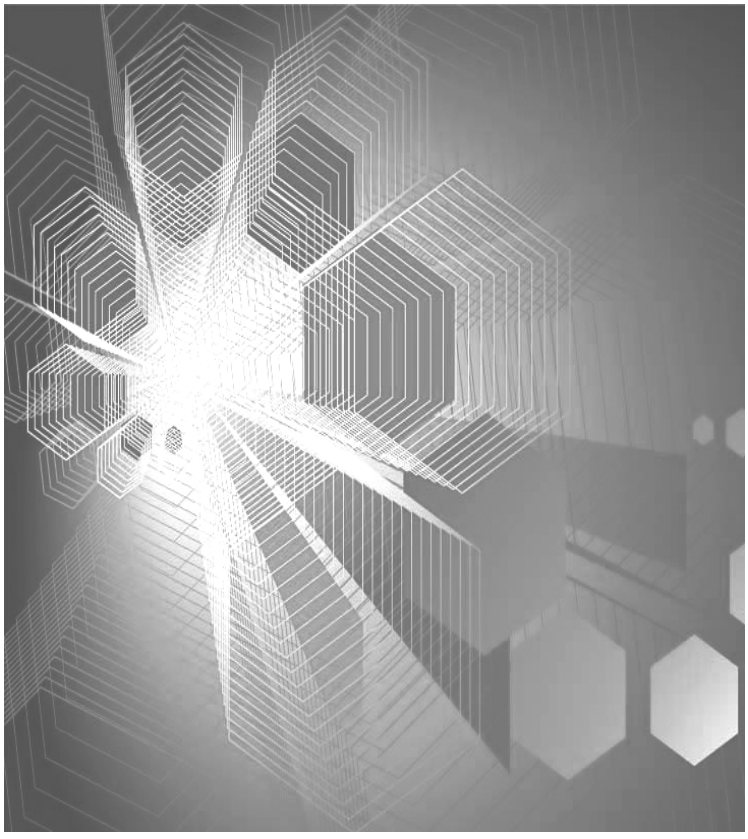
i	$a_{ij}, j=1,\cdots,4$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

对于 f_{21}、f_{22} 和 f_{23}，m 的值分别为 5、7、10，并且局部最优点的个数分别是 5、7 和 10。

$$x_{\text{local\_opt}} \approx a_i$$

$$f\left(x_{\text{local\_opt}}\right) \approx \frac{1}{c_i}$$

附录 B　约束优化问题



测试问题 1（T1）：

Minimize

$$f(x)=(x_1-2)^2+(x_2-1)^2$$

Subject to：

$$x_1=2x_2-1,\ \ x^2/4+x_2^2-1\leqslant 0$$

最优解 $f^*=1.3934651$

测试问题 2（T2）：

Minimize

$$f(x)=(x_1-10)^3+(x_2-20)^3$$

Subject to：

$$\begin{aligned}&100-(x_1-5)^2-(x_2-5)^2\leqslant 0\\&(x_1-6)^2+(x_2-5)^2-82.81\leqslant 0\\&13\leqslant x_1\leqslant 100,\ \ 0\leqslant x_2\leqslant 100\end{aligned}$$

最优解 $f^*=-6961.81381$

测试问题 3（T3）：

Minimize

$$f(x)=(x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7$$

Subject to：

$$\begin{aligned}&-127+2x_1^2+3x_2^4+x_3+4x_4^2+5x_5\leqslant 0\\&-282+7x_1+3x_2+10x_3^2+x_4-x_5\leqslant 0\\&-196+23x_1+x_2^2+6x_6^2-8x_7\leqslant 0\\&4x_1^2+x_2^2-3x_1x_2+2x_3^2+5x_6-11x_7\leqslant 0\\&-10\leqslant x_i\leqslant 10,\ \ i=1,\mathbf{L},7\end{aligned}$$

最优解 $f^*=680.630057$

测试问题 4（T4）：

Minimize

$$f(x)=5.3578547x_3^2+0.8356891x_1x_5+37.293239x_1-40792.141$$

Subject to：

$$0\leqslant 85.334407+0056858T_1+T_2x_1x_4-0.0022053x_3x_5\leqslant 92$$
$$90\leqslant 80.51249+0.0071317x_2x_5+0.0029955x_1x_2+0.0021813x_3^2\leqslant 110$$
$$20\leqslant 9.300961+0.0047026x_3x_5+0.0012547x_1x_3+0.0019085x_3x_4\leqslant 25$$
$$78\leqslant x_1\leqslant 102,\ 33\leqslant x_2\leqslant 45$$
$$27\leqslant x_i\leqslant 45\quad i=3,4,5$$

其中，$T_1=x_2x_5$，$T_2=0.0006262$。

最优解 $f^*=-30665.538$

测试问题 5（T5）：

$$f(x)=5x_1+5x_2+5x_3+5x_4-5\sum_{i=1}^{4}x_i^2-\sum_{i=5}^{13}x_i$$

Subject to：

$$2x_1+2x_2+x_{10}+x_{11}\leqslant 10$$
$$2x_1+2x_3+x_{10}+x_{12}\leqslant 10$$
$$2x_2+2x_3+x_{11}+x_{12}\leqslant 10$$
$$-8x_1+x_{10}\leqslant 0$$
$$-8x_2+x_{11}\leqslant 0$$
$$-8x_3+x_{12}\leqslant 0$$
$$-2x_4-x_5+x_{10}\leqslant 0$$
$$-2x_6-x_7+x_{11}\leqslant 0$$
$$-2x_8-x_9+x_{13}\leqslant 0$$
$$0\leqslant x_i\leqslant 1\quad i=1,\mathbf{L},9$$
$$0\leqslant x_i\leqslant 100\quad i=10,11,12$$
$$0\leqslant x_{13}\leqslant 1$$

最优解 $f^*=-15$

测试问题 6（T6）：

$$f(x) = x_1 + x_2 + x_3$$

Subject to：

$$\begin{aligned}
&1 - 0.0025(x_4 + x_6) \geqslant 0 \\
&1 - 0.0025(x_5 + x_7 - x_4) \geqslant 0 \\
&1 - 0.01(x_8 - x_5) \geqslant 0 \\
&x_1 x_6 - 833.33252 x_4 - 100 x_1 + 83333.33 \geqslant 0 \\
&x_2 x_7 - 1250 x_5 - x_2 x_4 + 1250 x_4 \geqslant 0 \\
&x_3 x_8 - 1250000 - x_3 x_5 + 2500 x_5 \geqslant 0 \\
&100 \leqslant x_1 \leqslant 10000 \\
&1000 \leqslant x_i \leqslant 10000 \quad i = 2,3 \\
&10 \leqslant x_i \leqslant 1000 \quad i = 4, \mathbf{L}, 8
\end{aligned}$$

最优解 $f^* = 7049.330923$

测试问题 7（T7）：

$$f(x) = \mathrm{e}^{x_1 x_2 x_3 x_4 x_5}$$

Subject to：

$$\begin{aligned}
&x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 = 10 \\
&x_2 x_3 - 5 x_4 x_5 = 0 \\
&x_1^3 + x_2^3 = -1 \\
&-2.3 \leqslant x_i \leqslant 2.3 \quad i = 1,2 \\
&-3.2 \leqslant x_i \leqslant 3.2 \quad i = 3,4,5
\end{aligned}$$

最优解 $f^* = 0.0539498478$

测试问题 8（T8）：

$$\begin{aligned}
f(x) = {} & x_1^2 + x_2^2 + x_1 x_2 - 14 x_1 - 16 x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\
& + (x_6 - 1)^2 + 5 x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45
\end{aligned}$$

Subject to：

$$
\begin{aligned}
&105-4x_1-5x_2+3x_7-9x_8 \geqslant 0\\
&-10x_1+8x_2+17x_7-2x_8 \geqslant 0\\
&8x_1-2x_2-5x_9+2x_{10}+12 \geqslant 0\\
&-3(x_1-2)^2-4(x_2-3)^2-2x_3^2+7x_4+120 \geqslant 0\\
&-5x_1^2-8x_2-(x_3-6)^2+2x_4+40 \geqslant 0\\
&-x_1^2-2(x_2-2)^2+2x_1x_2-14x_5+6x_6 \geqslant 0\\
&-0.5(x_1-8)^2-2(x_2-4)^2-3x_5^2+x_6+30 \geqslant 0\\
&3x_1-6x_2-12(x_9-8)^2+7x_{10} \geqslant 0\\
&-10.0 \leqslant x_i \leqslant 10.0 \quad i=1,\mathbf{L},10
\end{aligned}
$$

最优解 $f^*=24.3062091$

参 考 文 献

[1] 袁亚湘, 孙文瑜. 最优化理论与方法. 北京：科学技术出版社, 2001.

[2] 王子若, 陈永昌. 优化计算方法. 北京：机械工业出版社, 1989.

[3] 陈宝林. 最优化理论与算法. 北京：清华大学出版社, 1989.

[4] Schwefel H P. Numerical Optimization for Computer Model, Chichester, UK: John Wiley,1981.

[5] Bremermann H J. Optimization through evolution and recombination, Washington, D C, Spartan: Yovits M C, ed. Self-Organizing Systems,1962.

[6] Friedberg R M. A learning machine: Part 1, IBM Journal, 2(1): 2~13. 1958.

[7] Friedberg R M, Dunham B, North J H. A learning machine: Part 2, IBM Journal, 3(7):282~287. 1958.

[8] Box J E P. Evolutionary operation: a method for increasing industrial productivity, Appl Statistics, 6(2): 81~101. 1957.

[9] Holland J H. Outline for a logical theory of adaptive systems, J Assoc Compute Mach, (3):297~314. 1962.

[10] Rechenberg I. Cybernetic Solution Path of an Experimental Problem, UK: Farnborough, 1965.

[11] Schwefel H P. Pojekt MHD-Staustrahlrohr: Experimentelle Optimierung einer Zweiphasenduse. TeilI. Berlin, Germany: AEG Forschungsistitut, 1968.

[12] Fogel L J. Autonomous automata. Ind Res, (4):14~19. 1962.

[13] Back T, Hoffmeister F, Schwefel H P. Applications of evolutionary algorithms. Report of Systems Analysis Research Group SYS-2/92. University of Dormund, 1993.

[14] Dasgupte D, Michalewicz Z. Evolutionary Algorithms in Engineering Applications, Berlin: Springer, 1997.

[15] Mitsuo Gen, Runwei Cheng. Genetic Algorithms and Engineering Design. New York: John and Sons, 1996.

[16] Holland J H. Adaptation in Natural and Artificial Systems, Ann Arbor. MI: University of Michigan Press, 1975.

[17] Holland J H, Reitman J S. Cognitive system based on adaptive algorithms, In: Waterman D A, Hayes-Roth F, eds. Pattern-directed Inference Systems. New York: Academic, 1978.

[18] De Jong K A. An analysis of the behavior of a class of genetic adaptive systems, [Ph D dissertation] Michigan: University of Michigan, 1976.

[19] De Jong K A. On using genetic algorithms to search program space, In: Grefenstette J J, ed. Proceedings of the Second Int Conf on Gas and their Applications Hillsdale, NJ: Lawrence Erlbaum, 210~216. 1987.

[20] De Jong K A are genetic algorithms function optimizers, In: Manner R, Manderick ,eds.Parallel Problem Solving from Nature-2 Amsterdam. The Netherlands:Elsevier, 3~13. 1992.

[21] De Jong K A. Genetic algorithms are NOT Function optimizers, In: Whitley LD, ed. Foundations of Genetic algorithms-2. San Mateo, CA: Morgan Kaufmann, 5~17. 1993.

[22] Goldberg D E. Genetic algorithms and rule learning in dynamic system control, IN: Grefenstette J J, ed. Proc 1st Int Conf on genetic algorithms and their Applications. Hillsdale, NJ: Lawrence Erlbaum, 1985.

[23] Goldberg D E. Genetic algorithms in Search, Optimization and Machine Learning, Reading. MA: Addison-Wesley, 1989.

[24] Goldberg D E. the theory of virtual alphabets, Lecture Notes in computer Science, Vol 496. Berlin, Germany: Springer, 13~22. 1991.

[25] Goldberg D E, Deb K, Clark J H. Genetic algorithms, nosie, and the sizing of populations, Complex Syst, (6):333~362. 1992.

[26] Goldberg D E, Deb K, Kargupta H, et al. Rapid, accurate optimization of difficult problem using fast messy genetic algorithms, In: Forrest S, ed. Proc 5th Int conf on Genetic Algorithms. San Mateo, CA:Morgan Kaufmann, 56~64. 1993.

[27] Fogel L J. On the organization of intellect [Ph D Dissertation]. Los angeles: University of California, 1964.

[28] Burgin G H. On playing two-person zero-sum games against nonminimax players, IEEE trans Syst Sci Cybern, SSC-5(4):369~370. 1969.

[29] Burgin G H. Systems identification by quasilinearization and evolutionary programming, J Cybern, 3(2):56~75. 1973.

[30] Atmar J W. Speculation on the evolution of intelligence and its possible realization in machine, [Ph D dissertation]. Las Cruces: New Mexico State University,1976.

[31] Fogel D B. Evolutionary Programming for training Neural Networks, Washington D C: International Joint Conf on Neural Networks, 1990.

[32] Fogel D B. Evolving neural networks, Biol Cybern, (63):487~493. 1990.

[33] Saravanan N, Fogel D B. Evolving neurocontrollers using evolutionary programming, In: Fogel D B, ed. Proc of IEEE Frist Int Conf on Evolutionary Computation. Orlando: IEEE Press, 217~222. 1994.

[34] Rechenberg I. Evolutionsstrategies: Optimierung technischer Systeme nach Prinzipien der Biologischen Evolution, Stuttgart, Germany: Frommann-Holzboog, 1973.

[35] Rechenberg I Evolutionsstrategie’94, Volume 1 of Werkstatt Bionik and Evolutionstechnik. Stuttgart, Germany: Frommann-Holzboog, 1994.

[36] Schwefel H P. Evolutionsstrategie and numerische Optimierung: [Ph D Dissertation]. Berlin: Technische Universitat Berlin, 1975.

[37] Schwefel H P. Evolution and Optimum Seeking. New York: (Sixth-generation Computer technology Series) Wiley, 1995.

[38] 康立山. 演化计算：数值计算与计算机应用, 3: 173~179. 1995.

[39] Back T, Hoffmerister F, Schwelfel H. P. A survey of Evolution Strategies, In: Proceedings of the Fourth International Conference on Genetic algorithms. Morgan Kaufmann Publishers. San Mateo. CA, 1991.

[40] Davis L. Adapting Operator Probabilities in Genetic Algorithms, In: Proceedings of the Third International Conference on Genetic algorithms. Morgan Kaufmann Publishers San Mateo. CA, 1989.

[41] Bake J. E. Reducing Bias and Inefficiency in Selection Algorithms, In: Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.

[42] Fogel D. B. (Editer), IEEE Transactions on Neural Networks, Special issue on Evolutionary Computation. Vol.1, No.1:77~97. 1993.

[43] Bake J. E. Adaptive Selection Methods for genetic algorithms, In: Proceedings of the first International Conference on genetic algorithms. Lawrence Erlbaum Associates, Hillsdale, NJ, 1985.

[44] Vignaux G. A, Michalewicz Z. Genetic Algorithms for the Transportation Problem, In: Proceedings of the 4th International Symposium on Methodologies for Intelligent Systems, North-Holland, Amsterdam, 1989.

[45] De Jong. K. A. An Analysis of Behavior of a Class of Genetic Adaptive Systems, [Doctoral dissertation], University of Michigan, Dissertation Abstract International, 36(10), 5140B. (University Microfilms No 76~81). 1993.

[46] Brindle A. Genetic Algorithms for Function Optimization, [Doctoral Dissertation], University of Alberta. Edmonton. 1991.

[47] Goldberg D. E, Segrest P. Finite Markov Chain analysis of genetic Algorithms, In: Proceedings of the Second Internaltional Conference on genetic algorithms. Lawrence Erlbaum Associates, Hillsdale, NJ, 1987.

[48] Rudolph G. Convergence analysis of Canonical Genetic Algorithms, IEEE transactions on Neural Networks. Special issue on evolutionary computation. Vol.5. No1. 1994.

[49] 周春光, 梁艳春. 计算智能. 长春: 吉林大学出版社, 2001.

[50] Szalas A, Michalewicz Z. Contractive Mapping Genetic Algorithms and their Convergence, Department of Computer Science. University of North Carolina at Charlotte, Technical Report006,1993.

[51] Fogel L. J., Owens A. J., Walsh M. J. Artificial Intelligence Through Simulate Evolution, Chichester. John Wiley. UK, 1996.

[52] Fogel D. B. Evolving artificial Intelligence. [PHD Thesis], University of California. San Diego. 1992.

[53] Fogel D. B. Evolutionary Computation: Toward A New Philosophy of Machine Intelligence, IEEE Press. Piscataway. NJ. 1995.

[54] Mcdonnell J R, Reynolds R G, Fogel D. B. (Editer), Proceedings of the Fourth Annual Conference on Evolutionary Programming. The MIT Press, 1995.

[55] Sebald A V, Fogel L. J. Proceedings of the Third Annual Conference on Evolutionary Programming. San Diego, CA, World Scientific. 1994.

[56] X. Yao, Y. Liu and G. Lin. Evolutionary Programming Made Faster, IEEE Transactions on Evolutionary Computation,Vol.3 No.2: 82~102.1999.

[57] X. Yao and Y. Liu, Scaling up Evolutionary Programming Algorithms, Proc. of the 7th Annual Conference on Evolutionary Programming:103~112.1998.

[58] X. Yao, G. Lin and Y. Liu. An Analysis of Evolutionary Algorithms Based on Neighborhood and Step Size, Proc. of the 6th Annual Conference on Evolutionary Programming, Lecture Notes in Computer Science Vol. 1213, Springer-Verlag.1997.

[59] Yoshiyuki Matsumura, Kazuhiro Ohkura and Kanji Ueda. Evolutionary Programming with Non-Coding Segments for Real-valued Function Optimization, Proc. of IEEE International Conference on Systems, Man and Cybernetics (SMC'99), vol.4, 1999.

[60] K. Ohkura, Y. Matsumura and K. Ueda. Robust Evolution Strategies, Applied Intelligence, Kluwer Academic Publishers, accepted for publication. 2002.

[61] Yoshiyuki Matsumura, Kazuhiro Ohkura and Kanji Ueda (2001). Evolutionary Dynamics of Evolutionary Programming in Noisy Environment. Proceedings of Congress on Evolutionary Computation (CEC2001) , 5.2001.

[62] Rudolph G.. Self-adaptation and Global Convergence: a Counter-example, Proc. of Cong. On Evol. Comp. 1999.

[63] Rudolph G.. Self-Adaptive Mutations May Lead to Premature Convergence. Proceedings of Congress on Evolutionary Computation (CEC2001), 2001.

[64] Yong Liu, Xin Yao, Qiangfu Zhao and Tetsuya Higuchi. Scaling Up Fast Evolutionary Programming with Cooperative Coevolution, Proc. of the 2001 IEEE Congress on Evolutionary Computation Seoul, Korea. 2001.

[65] R C Eberhart, J Kennedy. A new optimizer using particle swarm theory, Proceedings of the sixth International Symposium on Micro Machine and Human Science. Nagoya Japan. 39~43. 1995.

[66] J Kennedy, R C Eberhart. Particle Swarm Optimization. Proc IEEE International Conference on Neural Networks. IEEE Service Center, Piscataway, NJ, IV: 1942~1948. 1995.

[67] M Clerc. TRIBES-Aparameter Free Particle Swarm Optimizer, http://clerc.maurice.free.fr/PSO 2002-08-10/2003-10-08.

[68] Hu Xiaohui, R C Eberhart. Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems. IEEE Congress on Evolutionary Computation, Honolulu, Hawaii USA, 2002.

[69] A Salman. Discrete Particle Swarm Optimization for Heterogeneous Task Assignment Problem. Proceedings of World Multiconference on Systemics, Cybernetics and Informatics (SCI 2001). 2001.

[70] M Clerc. Discrete Particle Swarm Optimization: A Fuzzy Combinatorial Black Box, http://clerc.maurice.free.fr/PSO/Fuzzy_Discrete_PSO/Fuzzy_DPSO.htm. 2000-04-01/2003-10-08.

[71] T Krink, J S Vesterstrom, J Riget. Particle Swarm Optimization with Spatial Particle Extension, Proceeding of the IEEE Congress on Evolutionary Computation, Honolulu, Hawaii USA, 2002.

[72] Hirotaka, Yoshida, Kenichi. A particle Swarm Optimization for Reactive Power and Voltage Control Considering Voltage Stability, IEEE International Conference on Intelligent System Applications to Power Systems, Rio de Janeiro, 1999.

[73] M S Voss, Xin Feng. Arma Model Selection Using Particle Swarm Optimization and Aic Criteria, 15th Triennial World Congress, Barcelona Spain, 2002.

[74] K E Parsopoulos, M N Vrahatis. Particle Swarm Optimization Method in Multiobjective Problems, Proceedings of the 2002 ACM Symposium on Applied Computing, 603~607. 2002.

[75] Van den Bergh, F Engelbrecht A P. Cooperative Learning in Neural Networks using Particle Swarm Optimizers, South Africam Computer Journal, 26: 84~90. 2000.

[76] 李爱国, 覃征, 鲍复民, 贺升平. 粒子群优化方法. 计算机工程与应用,（21）:38. 2002.

[77] 徐海，刘石，马勇. 基于改进粒子群游优化的模糊逻辑系统自学习算法. 计算机工程与应用 7:4~7. 2000.

[78] K E Parsopoulos, M N Vrahatis. Recent approaches to global optimization problems through Particle Swarm Optimization. Natural Computing 1: 235~306. 2002.

[79] J. Kennedy, and R. C. Eberhart. A discrete binary version of the particle swarm algorithm, Proceedings of the 1997 Conf. on Systems, Man, and Cybernetics, IEEE Service Center, Piscataway, NJ, 1997.

[80] Vesterstrøm J., S. and Riget J. Particle Swarms: Extensions for Improved Local, Multi-modal, and Dynamic Search in Numerical Optimization, [Master's thesis], 2002.http://www.evalife.dk/publications/

[81] P. J. Angeline. Using Selection to Improve Particle Swarm Optimization, In Proceedings of IJCNN’99, Washington, USA, July, 1999.

[82] M. Løvbjerg, T. K. Rasmussen, and T. Krink. Hybrid Particle Swarm Optimiser with Breeding and Subpopulations. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), San Francisco, USA, July, 2001.

[83] R C Eberhart, Yuhui Shi. Comparison between Genetic Algorithms and Particle Swarm Optimization, Proceedings of the Seventh Annual Conference on Evolutionary Programming. Springer Verlag. Ⅶ:611~618. 1998.

[84] P J Angeline. Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences, Proceedings of the Seventh Annual Conference on Evolutionary Programming, Springer Verlag. Ⅶ:601~610. 1998.

[85] M Clerc, J Kennedy. The Particle Swarm: Explosion, Stability, and Convergence in a Multi-Dimensional Complex Space, IEEE Transactions on Evolutionary Computation, Vol 6, No. 1: 58~73.2002.

[86] Maurice Clerc. Some math about Particle Swarm Optimization, http://clerc.maurice.free.fr/pso/ 2002-08-10/2003-10-08.

[87] Frans Van den Bergh. An analysis of Particle Swarm Optimizers: [Ph D dissertation]. Pretoria:University of Pretoria, 2001.

[88] Maurice Clerc. Particle Swarm Optimization and Information, http://clerc.maurice.free.fr/pso/ 2002-08-10/2003-10-08.

[89] F van den Bergh, A P Engelbrecht. Effects of Swarm Size on Cooperative Particle Swarm Optimisers, Proceedings of the Genetic and Evolutionary Computation Conference. San Francisco, USA. 2001.

[90] A Carlisle, G Dozier. Tracking Changing Extrema with Adaptive Particle Swarm Optimizer, Auburn University, Report No. Technical Report CSSE01-08, 2001.

[91] T M Blackwell. Swarms in Dynamic Environments, Genetic and Evolutionary Computation Conference, volume 2723 of LNCS. Springer, 2003.

[92] A Carlisle, G Dozier. Adapting Particle Swarm Optimization to Dynamic Environments, Proceedings, 2000ICAI, Las Vegas, NV, VolⅠ:429~434, 2000.

[93] K. E Parsopoulos, M. N Vrahatis. Particle swarm optimizer in noisy and continuously changing environments, Artificial Intelligence and Soft Computing IASTED/ACTA Press, Anaheim, CA, USA, 2001.

[94] P J Angeline. Tracking Extrema in Dynamic Environments, Proceedings of the 6th Int Conference on Evolutionary Programming, Ⅵ, 1997.

[95] R. Storn, K. Price. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous space Technical report, International Computer Science Institute, Berkley, 1995.

[96] Jakob Vesterstrom, Ren Thomsen. A Comparative Study of Differential Evolution,Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems, Proceedings of Congress on Evolutionary Computation (CEC2004), Poland, 2004.

[97] Abbass H., Sarker R., and Newton C. A Pareto frontier differential evolution approach for multiobjective optimization problems, In Proceedings of the Congress on Evolutionary Computation, volume2: 971~978. IEEE Service Centre. 2001.

[98] Krink T., Filipic B., and Fogel G. B. Noisy optimization problems-a particular challenge for differential evolution, In Proceedings of Congress on Evolutionary Computation (CEC2004), Poland 2004.

[99] Ricardo Landa, Becerra D, Carlos A. A Cultural Algorithm with differential Evolution to Solve Constrained Optimization Problems, Springer-Verlag, Lecture Notes in Artificial Intelligence Vol. 3315, Puebla, México, November 2004.

[100] Mezura Montes, et al. Simple Feasibility Rules and Differential Evolution for Constrained Optimization, In Proceedings of the Third Mexican International Conference on Artificial Intelligence, Springer Verlag, Lecture Notes in Artificial Intelligence Vol. 2972, April 2004.

[101] K. E. Parsopoulos, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, M. N. Vrahatis. Vector Evaluated Differential Evolution for Multiobjective Optimization http://www.math.upatras.gr/~dtas/papers/ ParsopoulosTPPV2004.pdf. 2004.

[102] Wen-Jun Zhang, Xiao-Feng Xie. DEPSO: Hybrid Particle Swarm with Differential Evolution Operator In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'2003), volume 4, 3816~3821. Washington DC, USA, IEEE, Octuber 2003.

[103] Kennedy J. Bare bones particle swarms, IEEE Swarm Intelligence Symposium :80～87, 2003.

[104] Richardson J T, Palmer M R, Liepins G, Hilliard M. Some Guidelines for Genetic Algorithms with Penalty Functions, Proceedings of Third International Conference on Genetic Algorithms. Morgan Kaufmann Publishers, San Mateo, CA. 1989.

[105] Powell D, Skolnick M M. Using Genetic Algorithms in Engineering Design Optimization with Non-linear Constraints, In: Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA. 1993.

[106] Michalewicz Z, Xiao J. Evolution of Paths in Evolutionary Planner/Navigator, Proceedings of 1995 International Workshop on Biologically Inspired Evolutionary Systems, Tokyo, Japan, 1995.

[107] Whitey D, Gordon V S, Mathias K. Lamarckian Evolution, the Baldwin Effect and Function Optimization, Proceedings of the Third International Conference on Parallel Problem Solving From Nature (PPSN), Springer-Verlag, New York,1994.

[108] Orvosh D, Davis L. Shall We Repair? Genetic Algorithms, Combinatorial Optimization, and Feasibility Constraints, In: Proceedings of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA. 1993.

[109] Homaifar A, Lai S H, Qi X. Constrained Optimization via Genetic Algorithms, Simulation. Vol.62.:242~254. 1994.

[110] Joines J A, Houck C R. On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Optimization Problems With GAs, Proceedings of the First IEEE Internaltional Conference on Evolutionary Computation, IEEE Service Center, Piscataway, NJ, Volume 2, Orlando, 27~29 June, 1994.

[111] Schaffer J D, et al. Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art, Proceedings of the International Workshop on Combinations Genetic Algorithms and Neural Networks. Baltimore, MD. June 6:1~37. 1992.

[112] Powell M J D. Variable Metric Methods for Constrained Optimization, Mathematical Programming: The State of the Art. Springer-Verlag, 1993.

[113] Robert G. Reynolds , Using Region-Schema to Solve Nonlinear Constraint Optimization Problems: a Cultural Algorithm Approach, Festschrift Conference in honor of John H. Holland,1999.

[114] Chan-Jin Chung and Robert G. Reynolds , The Use of Cultural Algorithms Support Self-adaptation in EP , Proc. Of Adaptive Distributed Parallel Computing Symposium , Dayton , Ohio , 2001.

[115] Chan-Jin Chung and Robert G. Reynolds, CAEP : An Evolution-based Tool for Real-Valued Function Optimization using Cultural Algorithms, International Journal on Artificial Intelligence Tools, Vol. 7, No.3 :1. 2002.

[116] Robert G. Reynolds and C. Chung, A Self-Adaptive Approach to Representation Shifts in Cultural Algorithms, Proceedings of IEEE International Conference on Evolutionary Computation, Nagoya, Japan. 2000.

[117] 李宁, 孙德宝, 邹彤, 秦元庆, 尉宇. 基于差分方程的 PSO 算法粒子运动轨迹分析. 计算机学报. Vol. 29, No.11, Nov. 2006.